BAB IV
ANALISA DAN PERANCANGAN

4.1 Analisa *Source Detection*

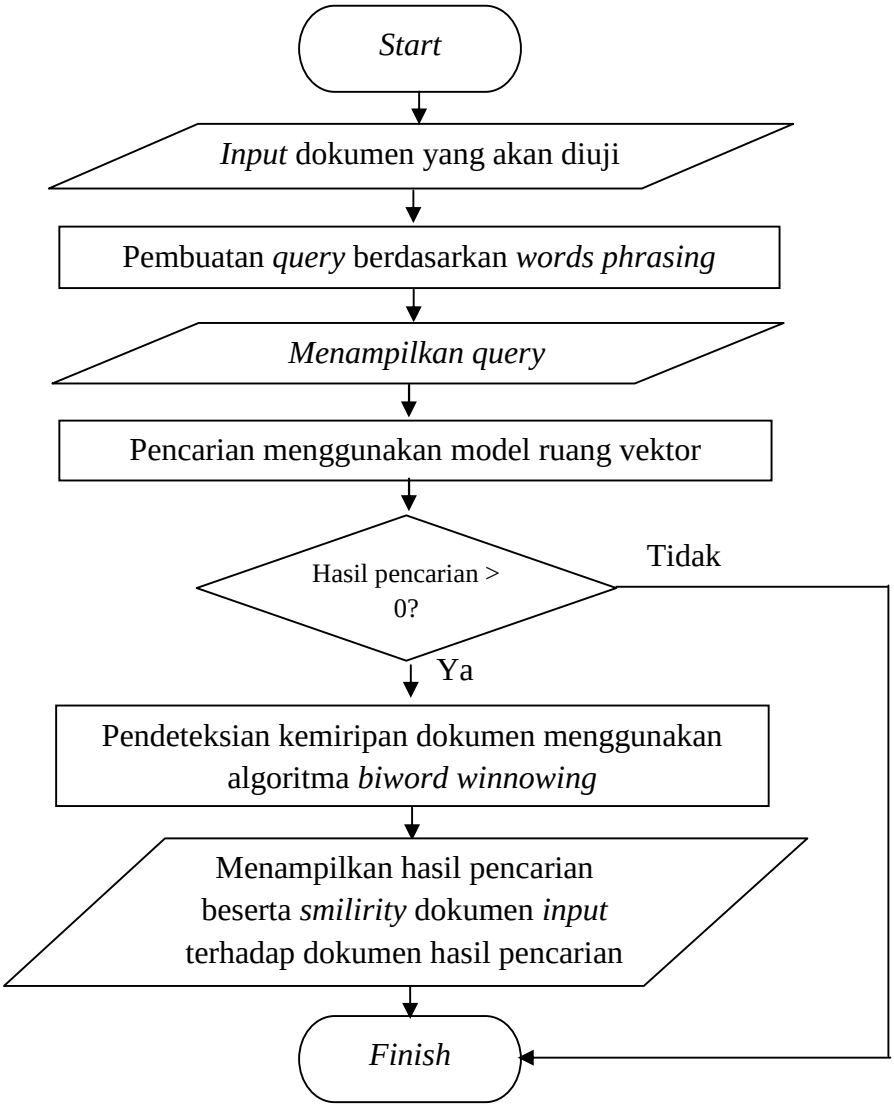
Pada tahapan ini akan menjelaskan tentang gambaran secara garis besar terhadap proses yang dilakukan aplikasi *source detection* pada kasus dokumen plagiarisme berdasarkan *words phrasing* dengan menggunakan model ruang vektor. Proses yang ada pada aplikasi ini terdiri atas tiga proses utama yang saling berhubungan satu sama lain, diantaranya: pembuatan *query*, mesin pencari dengan model ruang vektor dan proses pendeteksian kemiripan dokumen menggunakan algoritma *biword winnowing* dan *Jaccard Coefficient.*

Pada dasarnya setiap proses utama akan melakukan tahapan secara umum yaitu: *input*, proses dan *output*. Proses dimulai ketika *user* memasukkan sebuah dokumen teks yang di duga melakukan plagiarisme lalu di proses pada pembuatan *query*, pada tahapan pembuatan *query* dokumen teks akan di proses berdasarkan *words phrasing* sehingga akan menghasilkan *output* berupa *query*. Setelah selesai pada proses pembuatan *query*, proses selanjutnya adalah menggunakan *query* hasil dari proses sebelumnya sebagai *input* pada proses pencarian dengan model ruang vektor sehingga akan menghasilkan rangking pencarian berdasarkan tingkat kemiripan dokumen terhadap *query*. Hasil pencarian tersebut akan di evaluasi lagi kemiripannya terhadap dokumen yang di *input* dari awal pembuatan *query* menggunakan algoritma *biword winnowing* yang memiliki tahapan-tahapan yang cukup kompleks dalam menghitung kemiripan antar dua dokumen yang di proses.

Pada tahap menghitung kemiripan dokumen menggunakan algoritma *biword winnowing* tediri dari dua *input* yaitu: *input* awal dari proses pembuatan query kemudian *input* berdasarkan dokumen yang telah didapatkan dari hasil pencarian menggunakan model ruang vektor. Selanjutnya dokumen akan diproses berdasarkan tahapan yang dimiliki oleh algoritma pendeteksi plagiarisme. Tahapan tersebut diantaranya adalah tahap *preprocessing* dan tokenisasi. Proses

akan berlanjut dengan perhitungan tingkat *similarity* dokumen. Setelah proses utama dilakukan, selanjutnya aplikasi akan menghasilkan *output* informasi dokumen berupa hasil *similarity* dokumen dan kalimat yang telah diplagiasi.

Gambar 4.1 dibawah adalah *flowchart* yang akan menggambarkan proses-proses yang akan dilakukan secara keseluruhan:



Gambar 4.1 *Flowchart source detecction*

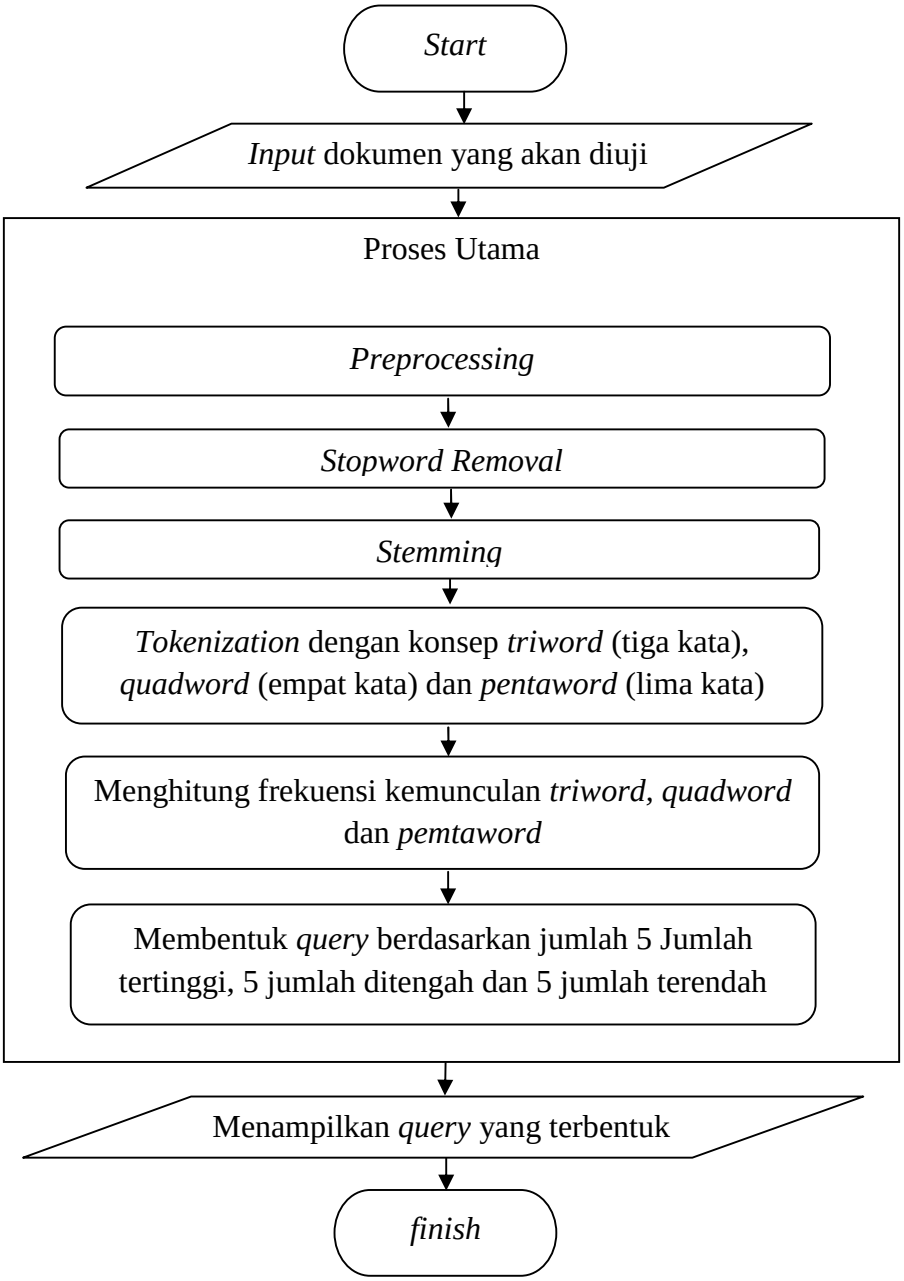
4.2 Analisa Pembuatan *Query*

Pada penelitian ini fokus utama penelitian terletak pada proses pembuatan *query* yang bertujuan mengekstrak isi dari sebuah dokumen teks menjadi *query* yang bias mewakili dari isi dokumen teks tersebut. Analisa pembentukan *query*

dari dokumen yang dicurigai plagiat berdasarkan *words phrasing* (frasa kata) dengan pendekatan *triword, quadword* dan *pentaword*. Adapun *query* yang akan dibentuk, dibedakan menjadi dua yaitu :

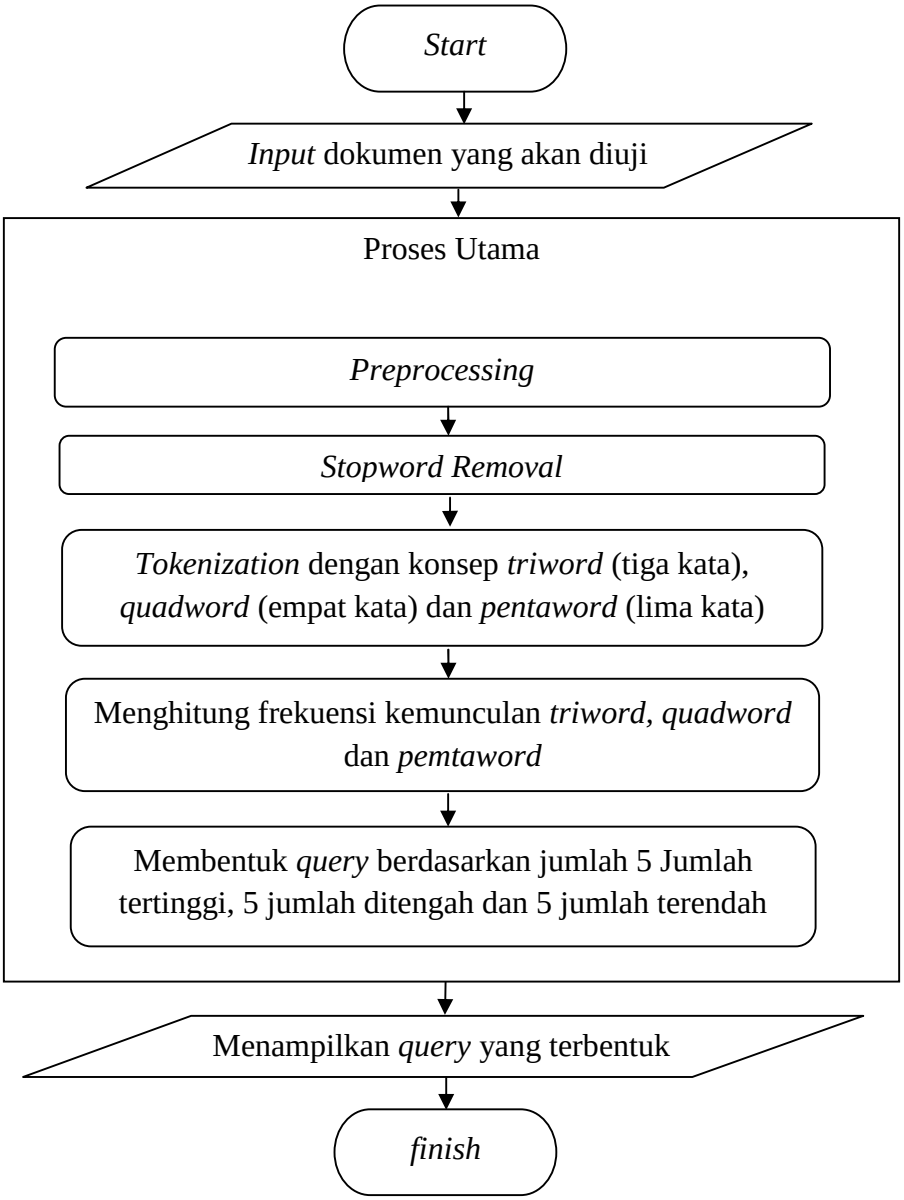
1. *Query* dengan menggunakan *stemming,* pada tahap ini akan dilakukan :
    a. Pembersihan teks dokumen.
    b. Menghilangkan angka pada dokumen.
    c. Melakukan pemotongan teks dokumen kedalam token tunggal, serta melakukan pembuangan *stopwords* dan *stemming,* kemudian menyusun token menjadi *triword, quadword* dan *pentaword*
    d. Melakukan perhitungan frekuensi kemunculan *triword, quadword* dan *pentaword* yang sama.
    e. Membentuk masing-masing *query* berdasarkan rangking frekuensi yaitu 5 *triword, quadword* dan *pentaword* dengan frekuensi tertinggi, 5 frekuensi terendah dan 5 frekuensi tengah.
2. *Query* dengan menggunakan tanpa *stemming,* pada tahap ini akan dilakukan :
    a. Pembersihan teks dokumen.
    b. Menghilangkan angka pada dokumen.
    c. Melakukan pemotongan teks dokumen kedalam token tunggal, serta melakukan pembuangan *stopwords* kemudian menyusun token menjadi *triword, quadword* dan *pentaword*
    d. Melakukan perhitungan frekuensi kemunculan *triword, quadword* dan *pentaword* yang sama.
    e. Membentuk masing-masing *query* berdasarkan rangking frekuensi yaitu 5 *triword, quadword* dan *pentaword* dengan frekuensi tertinggi, 5 frekuensi terendah dan 5 frekuensi tengah.

Gambar 4.2 adalah *flowchart* yang menggambarkan proses-proses yang dilakukan pada tahapan pembuatan *query* berdasarkan *words phrasing* dengan menggunakan *stemming* algoritma Nazief & Adriani:



Gambar 4.2 *Flowchart* pembuatan *query* menggunakan algoritma *stemming* Nazief Adriani.

Gambar 4.3 berikut adalah *flowchart* yang menggambarkan proses-proses yang dilakukan pada tahapan pembuatan *query* berdasarkan *words phrasing* tanpa menggunakan *stemming*:



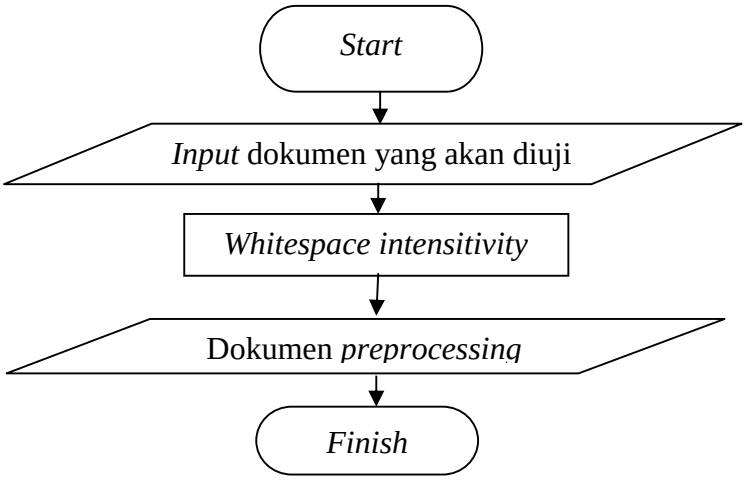
Gambar 4.3 *Flowchart* pembuatan *query* tanpa menggunakan algoritma *stemming*

Pada gambar 4.2 dan gambar 4.3 diatas dapat dilihati proses pembuatan *query* berdasarkan *words phrasing.* Proses-proses tersebut dapat dijelaskan sebagai berikut:

1. *Input* dokumen yang akan yang akan diuji kemiripannya pada aplikasi yang akan dibangun. Sehingga aplikasi akan memperoleh informasi dokumen yang akan diuji.

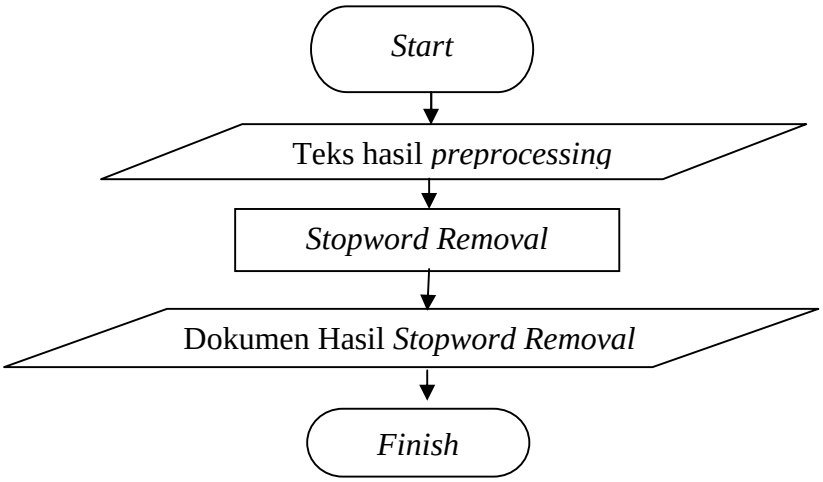
2. Dokumen yang telah dimasukkan akan diproses pada tahap *preprocessing*, yaitu menghilangkan karakter-karakter yang tidak relevan seperti membuang tanda baca, mengubah huruf besar menjadi huruf kecil, menghilangkan spasi dan membuang angka.

   Untuk lebih jelasnya, dapat dilihat pada gambar 4.4 berikut:



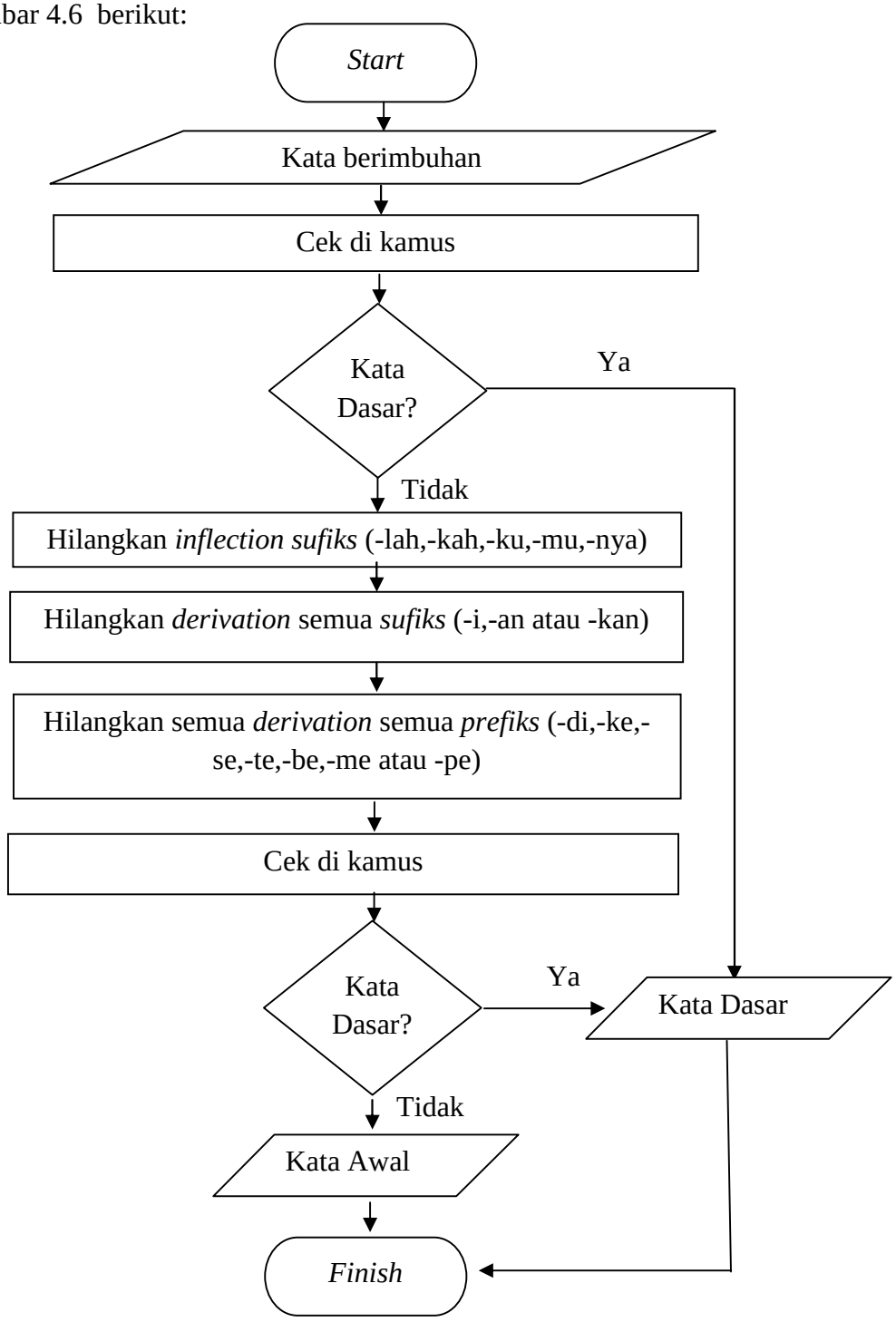
Gambar 4.4 *Flowchart Preprocessing* Dokumen

3. *Stopword Removal*, pada tahapan ini akan dilakukan pembuangan kata-kata yang dianggap selalu muncul dalam frekuensi tinggi yang tidak memberikan informasi secara tepat. Untuk lebih jelasnya dapat melihat gambar 4.5 berikut:



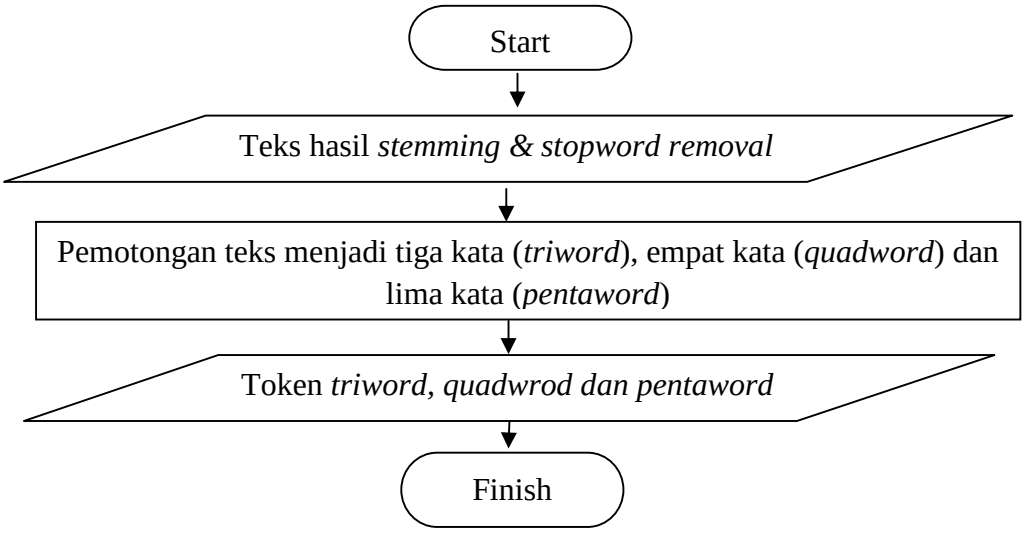
Gambar 4.5 *Flowchart Stopword Removal* (Pembuangan kata-kata yang tidak informatif)

4. Pada Tahapan ini akan dilakukan *stemming* atau pengembalian kata kedalam bentuk kata dasar menggunakan algoritma Nazief & Adriani, langkah-langkah penggunaan algoritma Nazief Adriani dapat dilihat pada gambar 4.6 berikut:



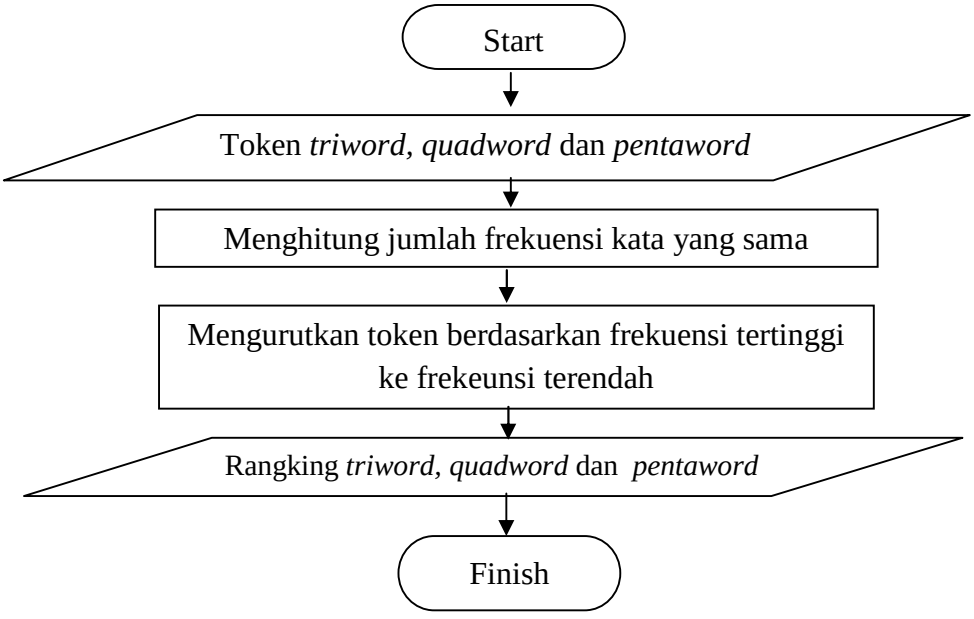
Gambar 4.6 *Flowchart* Algoritma *Stemming* Nazief Adriani

5. Pada tahapan ini akan dilakukan proses *words phrasing* dengan pendekatan *triword, quadword* dan *pentaword*. Untuk lebih jelasnya tergambar pada *flowchart* gambar 4.7 berikut:



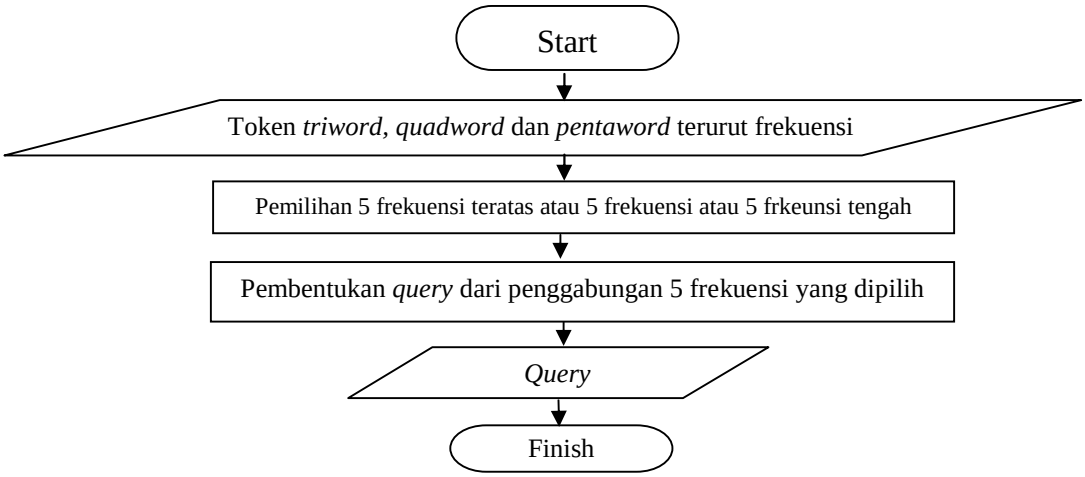
Gambar 4.7 *Flowchart* Pembentukan *triword,quadword* dan *pentaword*

6. Pada tahapan ini akan dilakukan perhitungan frekuensi kata yang sama berdasrkan *triword, quadword* dan *pentaword* yang terbentuk. Proses tersebut tergambar pada *flowchart* gambar 4.8 berikut:



Gambar 4.8 *Flowchart* Perhitungan frekuensi *triword,quadword* dan *pentaword*

7. Pada tahapan terakhir ini akan dilakukan proses pemilihan frekuensi kemunculan *triword, quadword* dan *pentaword* berdasarkan lima frekeunsi tertinggi, terendah dan lima frekuensi tengah.



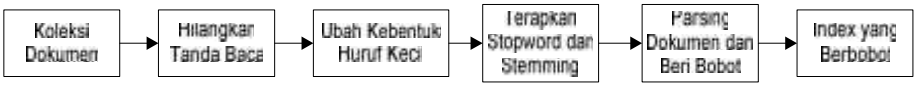
Gambar 4.9 *Flowchart* Pemilihan *triword, quadword* dan *pentaword* menjadi *query*

Gambar 4.9 menkelasakan tahapan-tahapan perhitungan kemunculan frekuensi kata. Proses pembuatan *query* selesai pada tahapan ini, hasil proses ini berupa *query* yang akan terus digunakan pada proses pencarian menggunakan model ruang vektor.

4.3 Analisa *Information Retrieval* dengan Model Ruang Vektor

Secara garis besar, ada tiga tahapan yang ditangani oleh sistem ini, yaitu melakukan preproses terhadap dokumen, melakukan preproses terhadap *query* pengguna dan menerapkan metode tertentu dalam hal ini menggunakan model ruang vektor untuk menghitung kedekatan (relevansi / *similirity*) antara dokumen dan *query* hasil pembuatan di proses sebelumnya. Adapun tiga tahapan tersebut, yaitu:

1. *Preprocessing* Dokumen



Gambar 4.10 Tahapan *preprocessing* dokumen

Berdasarkan gambar 4.10 tahapan *preprocessing* dokumen memiliki tahapan sebagai berikut:

a. Menyimpan dokumen kedalam koleksi dokumen

   Sebelum dilakukan tahapan preproses, semua dokumen yang akan dicari disimpan dalam sebuah koleksi dokumen. Adapun dokumen yang akan dijadikan koleksi dokumen adalah landasan teori kerja praktek dan tugas akhir dengan format pdf, doc dan txt.

b. Menghilangkan tanda baca pada dokumen

   Semua tanda baca yang ada pada koleksi dokumen akan dihilangkan.

c. Mengubah dokumen ke bentuk huruf kecil

   Tahapan preproses dokumen berikutnya adalah mengubah koleksi dokumen ke bentuk huruf kecil.

d. Menerapkan *stopword removal*

   Pada tahapan ini, setiap istilah yang tidak menggambarkan isi dari dokumen akan dihapus, seperti kata penghubung dan kata penunjuk yang mengacu pada koleksi *stopword*, misalnya : yang, ini, itu dan lain sebagainya.

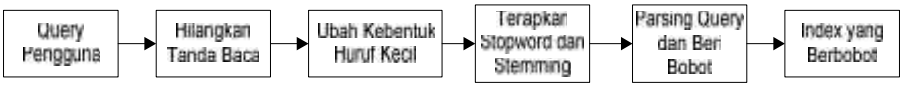
e. Menerapkan *stemming* (mengembalikan kata ke kata dasar)

   Dengan diterapkannya *stemming* diharapkan dapat meningkatkan performansi *information retrieval* yang akan dibangun. Adapun algoritma stemming yang akan akan digunakan yaitu algoritma Nazief & Adriani, lebih jelasnya dapat dilihat pada Gambar 4.6 untuk tahapan algoritma Nazief & Adriani.

f. Pembobotan setiap istilah pada dokumen

   Tahapan akhir dari preproses dokumen adalah pembobotan, dengan adanya pembobotan ini setiap kata akan di*parsing* dan dihitung jumlah kemunculannya.

2. *Preprocessing Query*



Gambar 4.11 Tahapan *preprocessing query*

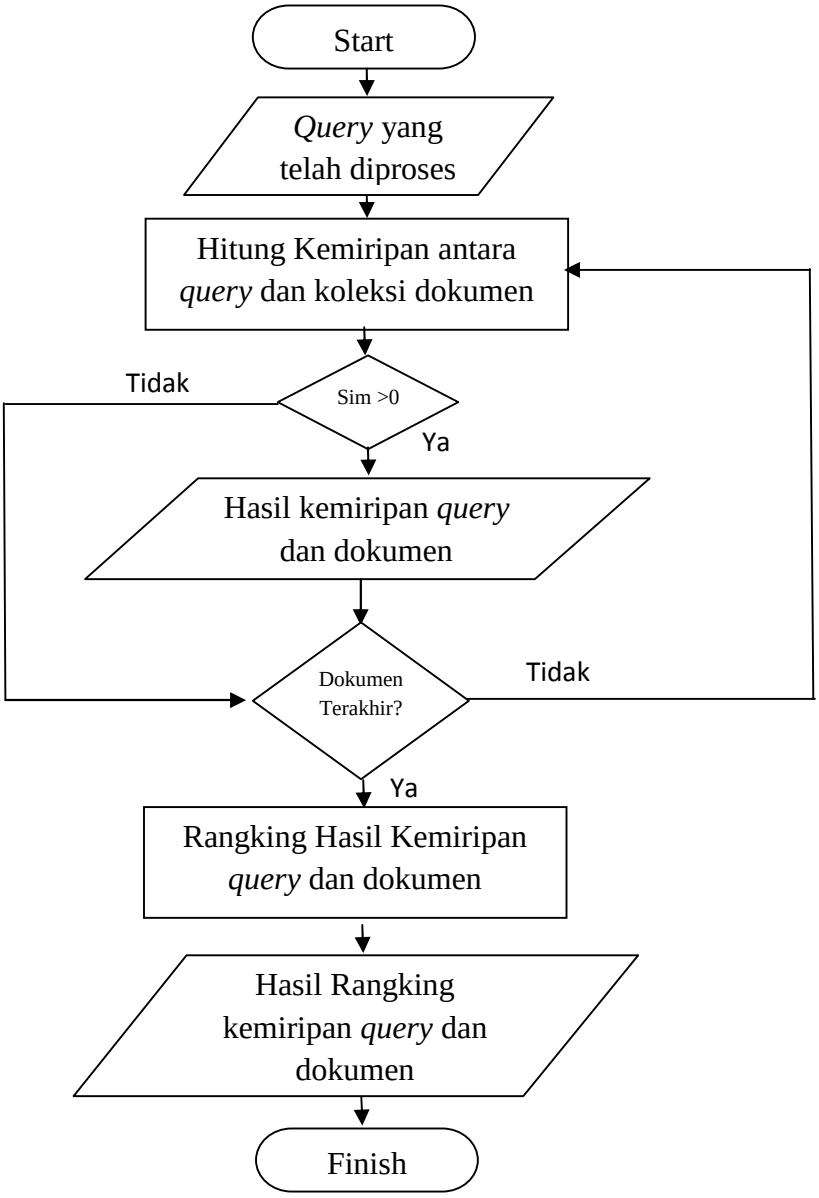
Berdasarkan gambar 4.11 *preprocessing query* memiliki tahapan sebagai berikut:

a. Menghilangkan tanda baca pada dokumen
b. Mengubah dokumen kebentuk huruf kecil
c. Menerapkan *stopword removal*
d. Menerapkan *stemming* (mengembalikan kata ke kata dasar)
e. Pembobotan setiap istilah *query*

   Pembobotan pada *query* ini mengacu dari hasil indexing pada preproses dokumen.

3. Penerapan Model Ruang Vektor

   Setelah pemberian bobot setiap istilah pada dokumen dan *query*, maka pada tahapan ini akan dilakukan perhitungan kemiripan antar *query* dan koleksi dokumen yang tersedia, *input* pada proses ini adalah *query* yang telah di preproses sehingga menjadi *query* yang tunggal. Tahapan tersebut dapat dilihat pada *flowchart* gambar 4.12 berikut:



Gambar 4.12 Penerapan Model Ruang Vektor

Berdasarkan gambar 4.12 penerapan model ruang vector memiliki tahapan sebagai berikut:

a. Hitung panjang vektor setiap dokumen dan query

   Sebelum dilakukan penghitungan relevansi dokumen dan *query*, setiap dokumen pada koleksi dan *query* akan dihitung panjang vektornya.

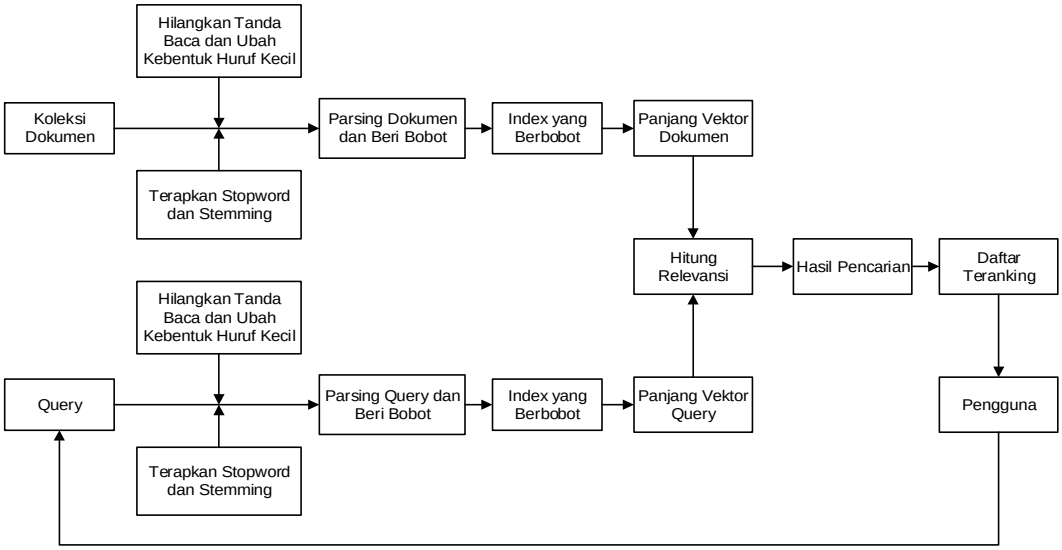
b. Hitung kedekatan (relevansi / similirity) antara dokumen dan *query* pengguna.

Setelah didapatkan panjang vektor setiap dokumen dan *query*, dilakukan penghitungan kedekatan query tersebut terhadap dokumen yang ada pada koleksi. Dari proses ini didapatkan relevansi / *similirity* yang akan dijadikan acuan dalam menentukan dokumen yang relevan sesuai *query* yang diinputkan.

c. Simpan hasil relevansi antara dokumen dan *query* dan lakukan perangkingan.

   Jika hasil dari perhitungan besar dari nol maka data perhitungan tersebut disimpan, jika tidak maka data perhitungan tidak disimpan.

Berdasarkan analisa dari tiga tahapan yang dilakukan oleh sistem temu balik informasi, maka dapat diilustrasikan seperti gambar 4.13 berikut.



Gambar 4.13 Tahapan dalam *Information Retrieval*

4.4 Algoritma *Winnowing* dengan pendekatan *biword*

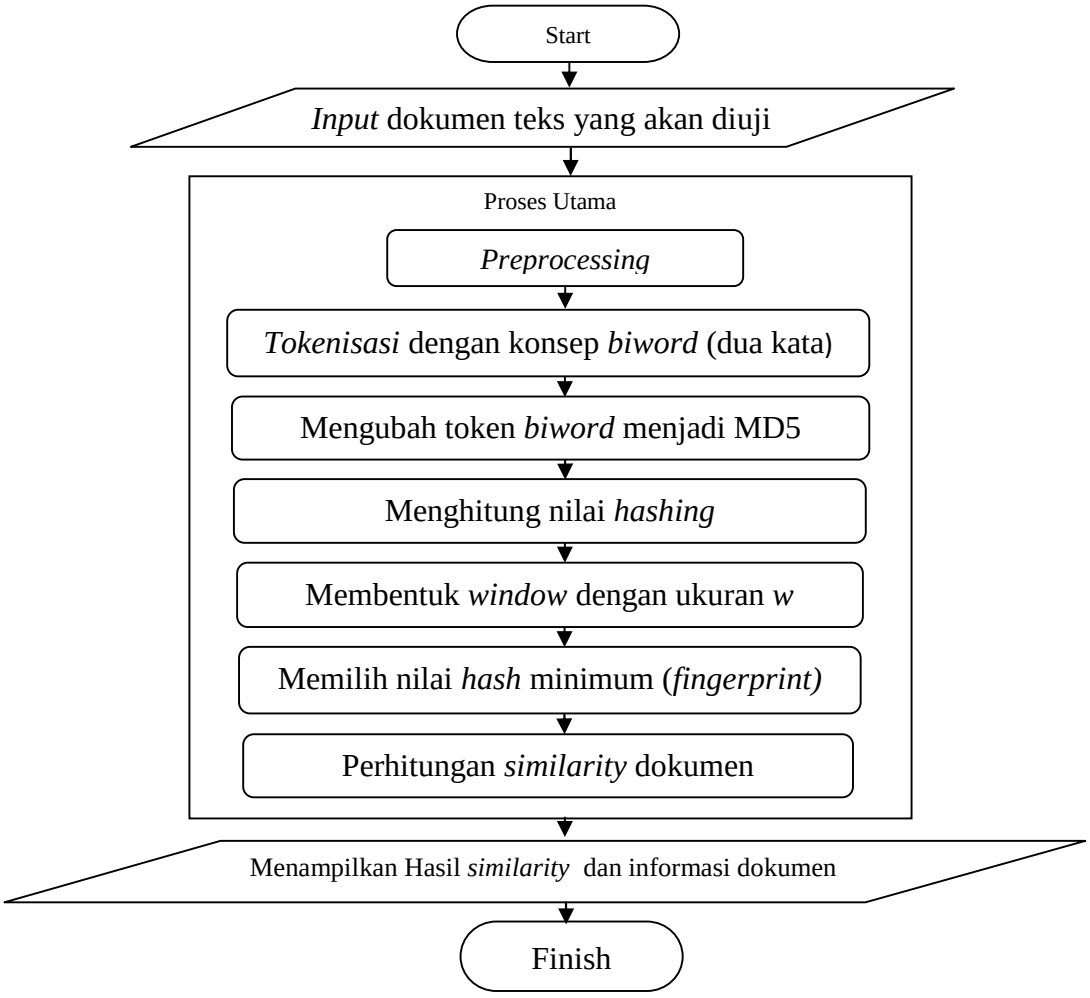
Pada penelitian ini dilakukan pengembangan algoritma *winnowing* dalam mendeteksi penjiplakan dokumen teks, yaitu dengan penerapan konsep *biword.* Algoritma *winnowing* yang biasanya menggunakan teknik *character-based* dalam proses tokenisasi dokumen, sekarang akan dilakukan menggunakan teknik *phrase-based.* Dengan demikian, akan terbentuk banyak frasa atau token *biword* dari masing-masing dokumen teks untuk perhitungan *similarity*. Konsep *biword* ini merupakan pendekatan *k-grams* untuk membentuk *substring* sepanjang k

karakter atau kata. Pendekatan k-grams inilah yang digunakan dalam membentuk token *biword.*

Secara garis besar ada beberapa tahap dalam melakukan pendeteksian plagiarisme dokumen menggunakan pendekatan *biword winnowing*, diantaranya:

1. Melakukan pembersihan teks.
2. Melakukan pemotongan teks kedalam bentuk *biword* kemudian dienkripsi menggunakan *MD5*
3. Menghitung nilai *hash*
4. Membentuk *window* dengan ukuran *w*
5. Mendapatkan nilai *fingerprint*
6. Menghitung kemiripan dokumen dari nilai *fingerprint* yang diperoleh.

Berikut adalah *flowchart* proses-proses yang dilakukan pada algoritma *biword winnowing* dalam mendeteksi penjiplakan dokumen teks:

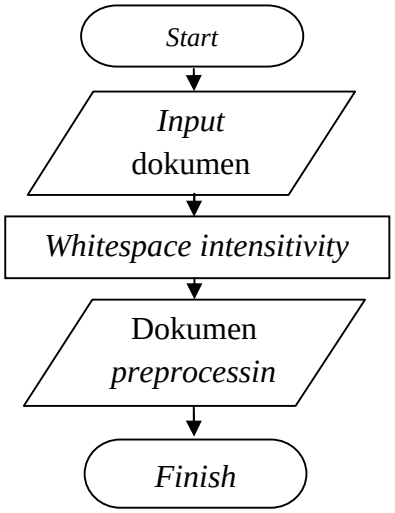


Gambar 4.14 *Flowchart* algoritma *winnowing* dengan pendekatan *biword*

Pada gambar 4.14 diatas dapat dilihat proses deteksi plagiarisme dokumen dengan menerapkan pendekatan *biword* (dua kata) pada algoritma *winnowing.* Proses-proses tersebut dapat dijelaskan sebagai berikut:

1. *Input* dokumen yang akan yang akan diuji kemiripannya pada aplikasi yang akan dibangun. Sehingga aplikasi akan memperoleh informasi dokumen yang akan diuji.
2. Dokumen yang telah dimasukkan akan diproses pada tahap *preprocessing,* yaitu menghilangkan karakter-karakter yang tidak relevan seperti membuang tanda baca, mengubah huruf besar menjadi huruf kecil dan menghilangkan spasi.

   Untuk lebih jelasnya, dapat dilihat pada *flowchart* gambar 4.15 berikut:

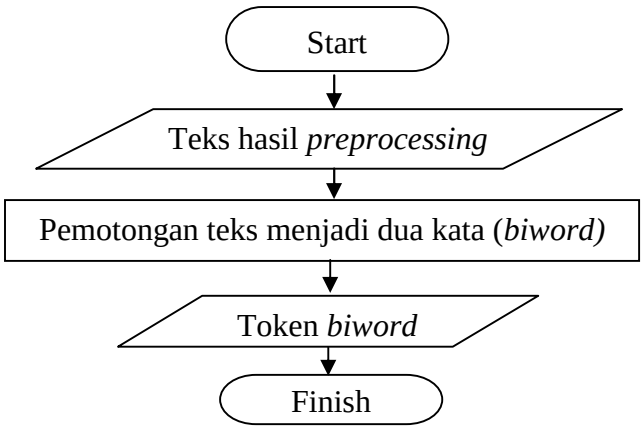


Gambar 4.15 *Flowchart* proses *preprocessing*

3. Tokenisasi dengan pendekatan *biword.*

   Setelah memperoleh dokumen *preprocessing,* selanjutnya dilakukan proses tokenisasi kata menjadi *biword.*

   Untuk lebih jelasnya dapat dilihat pada *flowchart* gambar 4.16 berikut:

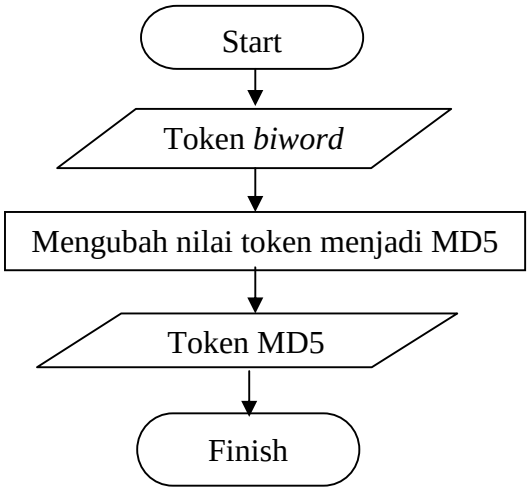


Gambar 4.16 *Flowchart* proses tokenisasi

4. Mengubah nilai token menjadi MD5

   Setelah mendapatkan token kata *biword*, selanjutnya akan dilakukan konversi mengubah nilai token *biword* menjadi nilai MD5. Hal ini bertujuan agar token tersebut memiliki panjang karakter yang sama yaitu 32 karakter.

   Untuk lebih jelasnya dapat dilihat pada *flowchart* gambar 4.17 berikut:

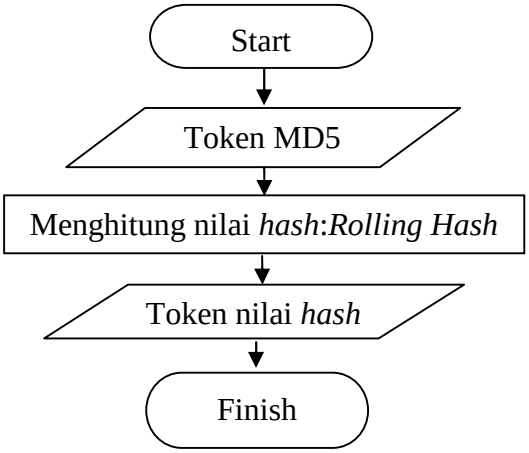


Gambar 4.17 *Flowchart* proses mendapatkan nilai MD5

5. Menghitung nilai *hash* masing-masing token.

   Token-token yang telah diubah menjadi MD5, selanjutnya akan diproses menggunakan persamaan *rolling hash* untuk mendapatkan nilai *hash* dokumen. Nilai *hash* ini nantinya akan dijadikan *fingerprint* dokumen.

Untuk lebih jelasnya dapat dilihat pada *flowchart* gambar 4.18 berikut:

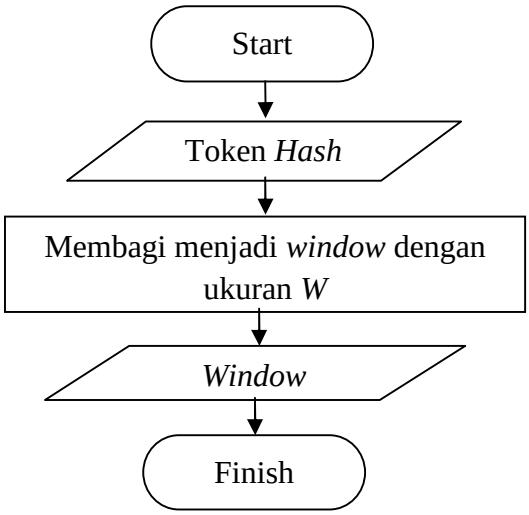


Gambar 4.18 *Flowchart* proses hitung nilai *hash*.

6. Membagi ke dalam beberapa *window*.

   Token-token yang telah diperoleh, akan dibagi dalam beberapa *window* dengan ukuran *w*. Ukuran *window* ditentukan oleh pengguna aplikasi.

   Gambar 4.19 berikut adalah *flowchart* pembentukan *window*:

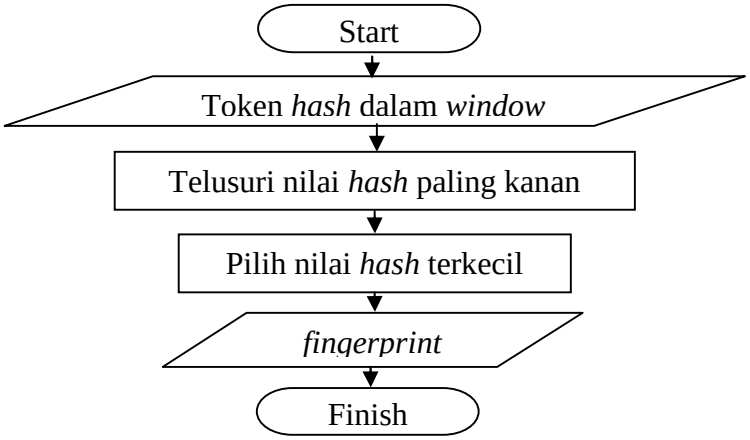


Gambar 4.19 *Flowchart* proses pembentukan *window*.

7. Mencari nilai *hash* minimum.

   Dari nilai-nilai *hash* yang telah dibentuk menggunakan persamaan *rolling hash*, selanjutnya akan ditelusuri nilai-nilai *hash* terkecil untuk dijadikan *fingerprint* dokumen. Penelusuran nilai *hash* terkecil adalah dimulai dari nilai *hash* yang paling kanan dalam suatu *window*.

Gambar 4.20 berikut adalah *flowchart* pencarian nilai *hash* terkecil:

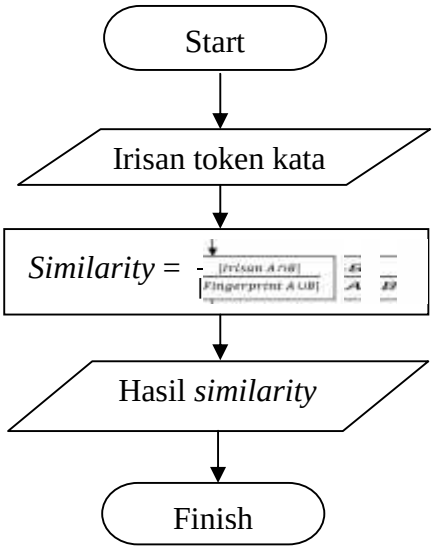


Gambar 4.20 *Flowchart* proses memilih *fingerprint.*

8. Perhitungan *similarity* dokumen

   Nilai *fingerprint* yang diperoleh akan digunakan untuk menghitung *similarity* dokumen. Proses perhitungan dilakukan menggunakan persamaan *jaccard coefficient.*

   Untuk lebih jelasnya dapat dilihat pada *flowchart* gambar 4.21 berikut:



Gambar 4.21 *Flowchart* proses hitung *similarity*

9. Selanjutnya akan diperoleh hasil dari proses utama berupa informasi dokumen yaitu nama dokumen, ukuran dokumen, waktu proses dan hasil *similarity* dokumen teks.

Untuk lebih jelasnya, berikut adalah contoh penerapan *source detection* dokumen:

Terdapat 1 buah dokumen uji dan 3 buah koleksi dokumen sebagai berikut:

a. Dokumen Uji:”Algoritma genetika yang memiliki kehandalan dalam menghasilkan output yang optimal dapat dimanfaatkan untuk menyelesaikan masalah tersebut dengan bantuan aplikasi. Pemilihan struktur data dan algoritma merupakan permasalahan yang kritis dalam disain sistem yang memungkinkan temu kembali dengan basis data berukuran besar secara efektif dan efisien”

b. Dokumen d_1 = Algoritma genetika yang memiliki kehandalan dalam menghasilkan output yang optimal dapat dimanfaatkan untuk menyelesaikan masalah tersebut dengan bantuan aplikasi

c. Dokumen d_2 = Sistem Pendukung Keputusan SPK adalah sekumpulan prosedur berbasis model untuk memproses data dan memberikan pertimbangan bagi manajer dalam mengambil keputusan

d. Dokumen d_3 = Pemilihan struktur data dan algoritma merupakan permasalahan yang kritis dalam disain sistem yang memungkinkan temu kembali dengan basis data berukuran besar secara efektif dan efisien

Tahapan-tahapan yang dilakukan adalah:

Pembuatan *Query*

1. *Whitespace Intensitivity* atau *preprocessing*, yaitu menghilangkan karakter yang tidak relevan seperti menghilangkan tanda baca dan mengubah huruf besar menjadi kecil serta menghilangkan angka. Sehingga terbentuk kalimat:

   Dokumen Uji:

   algoritma genetika yang memiliki kehandalan dalam menghasilkan output yang optimal dapat dimanfaatkan untuk menyelesaikan masalah tersebut dengan bantuan aplikasi pemilihan struktur data dan algoritma merupakan permasalahan yang kritis dalam disain sistem yang memungkinkan temu kembali dengan basis data berukuran besar secara efektif dan efisien

2. Tahapan selanjutnya adalah *stopword removal*, penghilangan kata-kata yang dianggap tidak relevan, sehingga akan terbentuk kalimat:

Dokumen Uji:

algoritma genetika kehandalan menghasilkan output optimal dimanfaatkan menyelesaikan bantuan aplikasi pemilihan struktur data algoritma permasalahan kritis disain sistem memungkinkan temu basis data berukuran efektif efisien

3. Mengembalikan kata kedalam bentuk kata dasar menggunakan algoritma Naizef & Adriani. Sehingga terbentuk kalimat berikut:

   Dokumen Uji:

   algoritma genetika kehandalan output optimal manfaat selesai bantu aplikasi pemilihan struktur data algoritma kritis disain sistem temu basis data berukuran efektif efisien

   Pada penggunaan algoritma Nazeif & Adriani masih terdapat banyak kesalahan dan kegagalan dalam mengembalikan kedalam bentuk kata dasar (Syahroni,2012)

4. Proses Berikutnya adalah mengubah kata kedalam bentuk *triword*, *quadword* dan *pentaword*. Pada contoh ini akan menggunakan bentuk *triword,* hasilnya sebagai berikut:

   [0] => algoritma genetika kehandalan
   [1] => genetika kehandalan output
   [2] => kehandalan output optimal
   [3] => output optimal manfaat
   [4] => optimal manfaat selesai
   [5] => manfaat selesai bantu
   [6] => selesai bantu aplikasi
   [7] => bantu aplikasi pemilihan
   [8] => aplikasi pemilihan struktur
   [9] => pemilihan struktur data
   [10] => struktur data algoritma
   [11] => data algoritma kritis
   [12] => algoritma kritis disain

[13] => kritis disain sistem

[14] => disain sistem temu

[15] => sistem temu basis

[16] => temu basis data

[17] => basis data berukuran

[18] => data berukuran efektif

5. Setelah mendapatkan bentuk *triword* atau *quadword* atau *pentaword* maka akan dilakukan perhitungan frekuensi kemunculan kata yang sama sebagai berikut:

   [kritis disain sistem] => 1

   [algoritma kritis disain] => 1

   [data algoritma kritis] => 1

   [disain sistem temu] => 1

   [sistem temu basis] => 1

   [data berukuran efektif] => 1

   [basis data berukuran] => 1

   [temu basis data] => 1

   [struktur data algoritma] => 1

   [pemilihan struktur data] => 1

   [output optimal manfaat] => 1

   [kehandalan output optimal] => 1

   [genetika kehandalan output] => 1

   [optimal manfaat selesai] => 1

   [manfaat selesai bantu] => 1

   [aplikasi pemilihan struktur] => 1

   [bantu aplikasi pemilihan] => 1

   [selesai bantu aplikasi] => 1

   [algoritma genetika kehandalan] => 1

6. Tahapan berikutnya melakukan pemilihan frasa berdasarkan frekuensinya yaitu: 5 frekuensi tertinggi, 5 frekuensi tengah dan 5 frekuensi terendah. Pada contoh ini akan memilih 5 frekuensi tertinggi berdasarkan urutan kata pada langkah sebelumnya, menghasilkan kata berikut:

   [kritis disain sistem] => 1

   [algoritma kritis disain] => 1

   [data algoritma kritis] => 1

   [disain sistem temu] => 1

   [sistem temu basis] => 1

7. Langkah terakhir pembuatan *query* adalah menggabungkan 5 pilihan frekuensi pada langkah sebelumnya menghasilkan sebuah *query* baru yang dianggap mewakili isi dari dokumen uji, hasilnya sebagai berikut:

   "kritis disain sistem algoritma data temu basis"

Pencarian Dengan Model Ruang Vektor

Tahapan Preproses Dokumen:

1. Menghilangan tanda baca

   a. Dokumen d_1 = Algoritma genetika yang memiliki kehandalan dalam menghasilkan output yang optimal dapat dimanfaatkan untuk menyelesaikan masalah tersebut dengan bantuan aplikasi

   b. Dokumen d_2 = Sistem Pendukung Keputusan SPK adalah sekumpulan prosedur berbasis model untuk memproses data dan memberikan pertimbangan bagi manajer dalam mengambil keputusan

   c. Dokumen d_3 = Pemilihan struktur data dan algoritma merupakan permasalahan yang kritis dalam disain sistem yang memungkinkan temu kembali dengan basis data berukuran besar secara efektif dan efisien

2. Mengubah istilah ke bentuk huruf kecil

   a. Dokumen d_1 = algoritma genetika yang memiliki kehandalan dalam menghasilkan output yang optimal dapat dimanfaatkan untuk menyelesaikan masalah tersebut dengan bantuan aplikasi

b. Dokumen d_2 = sistem pendukung keputusan spk adalah sekumpulan prosedur berbasis model untuk memproses data dan memberikan pertimbangan bagi manajer dalam mengambil keputusan
c. Dokumen d_3 = pemilihan struktur data dan algoritma merupakan permasalahan yang kritis dalam disain sistem yang memungkinkan temu kembali dengan basis data berukuran besar secara efektif dan efisien

3. Menerapkan *stopword removal*

   Adapun daftar *stop word* dari tiga contoh dokumen diatas adalah : yang, dalam, dapat, untuk, tersebut, dengan, adalah, dan, bagi, besar, secara.

   a. Dokumen d_1 = algoritma genetika memiliki kehandalan menghasilkan output optimal dimanfaatkan menyelesaikan masalah bantuan aplikasi
   b. Dokumen d_2 = sistem pendukung keputusan spk sekumpulan prosedur berbasis model memproses data memberikan pertimbangan bagi manajer mengambil keputusan
   c. Dokumen d_3 = pemilihan struktur data algoritma merupakan permasalahan kritis disain sistem memungkinkan temu kembali basis data berukuran efektif efisien

1. Menerapkan *stemming* (mengembalikan kata ke kata dasar)

   Adapun daftar *stemming* dari tiga contoh dokumen diatas adalah : milik, hasil, manfaat, selesai, bantu, dukung, putus, kumpul, basis, proses, beri, timbang, ambil, rupa, masalah, mungkin.

   a. Dokumen d_1 = algoritma genetika milik kehandalan hasil output optimal manfaat selesai masalah bantu aplikasi
   b. Dokumen d_2 = sistem dukung putus spk kumpul prosedur basis model proses data beri timbang bagi manajer ambil putus
   c. Dokumen d_3 = pemilihan struktur data algoritma rupa masalah kritis disain sistem mungkin temu kembali basis data berukuran efektif efisien

2. Pembobotan, setelah semua dokumen di*preprocessing* tiap *term* dipisah dan dimasukkan ke dalam tabel indexing.

   Dalam koleksi ini, terdapat tiga dokumen, sehingga diperoleh N = 3 dan berdasarkan rumus 2.3 maka untuk istilah algoritma dimana istilah algoritma

tersebut muncul pada pada 2 dokumen yaitu pada dokumen d_1 dan d_3 maka diperoleh df = 2, idf yang didapatkan adalah 0. 176.

Pembobotan untuk istilah algoritma dapat menggunakan rumus 2.9 sehingga untuk istilah algoritma diperoleh w (bobot) = 0. 176. Dengan penerapan rumus yang sama idf dan bobot setiap istilah selengkapnya dapat dilihat pada tabel 4.1.

Tabel 4.1 Hasil pembobotan index dokumen

No	Kata	TF			Df	Idf	Bobot		
		d1	d2	d3			d1	d2	d3
1	algoritma	1	0	1	2	0.1761	0.1761	0.0000	0.1761
2	ambil	0	1	0	1	0.4771	0.0000	0.4771	0.0000
3	aplikasi	1	0	0	1	0.4771	0.4771	0.0000	0.0000
4	bantu	1	0	0	1	0.4771	0.4771	0.0000	0.0000
5	basis	0	1	1	2	0.1761	0.0000	0.1761	0.1761
6	beri	0	1	0	1	0.4771	0.0000	0.4771	0.0000
7	berukuran	0	0	1	1	0.4771	0.0000	0.0000	0.4771
8	data	0	1	2	2	0.1761	0.0000	0.1761	0.3522
9	disain	0	0	1	1	0.4771	0.0000	0.0000	0.4771
10	dukung	0	1	0	1	0.4771	0.0000	0.4771	0.0000
11	efektif	0	0	1	1	0.4771	0.0000	0.0000	0.4771
12	efisien	0	0	1	1	0.4771	0.0000	0.0000	0.4771
13	genetika	1	0	0	1	0.4771	0.4771	0.0000	0.0000
14	kehandalan	1	0	0	1	0.4771	0.4771	0.0000	0.0000
15	kritis	0	0	1	1	0.4771	0.0000	0.0000	0.4771
16	kumpul	0	1	0	1	0.4771	0.0000	0.4771	0.0000
17	manajer	0	1	0	1	0.4771	0.0000	0.4771	0.0000
18	manfaat	1	0	0	1	0.4771	0.4771	0.0000	0.0000
19	model	0	1	0	1	0.4771	0.0000	0.4771	0.0000
20	optimal	1	0	0	1	0.4771	0.4771	0.0000	0.0000
21	output	1	0	0	1	0.4771	0.4771	0.0000	0.0000
22	pemilihan	0	0	1	1	0.4771	0.0000	0.0000	0.4771
23	prosedur	0	1	0	1	0.4771	0.0000	0.4771	0.0000
24	putus	0	2	0	1	0.4771	0.0000	0.9542	0.0000

Tabel 4.1 Hasil pembobotan index dokumen (lanjutan)

No	Kata	TF			Df	Idf	Bobot		
		d1	d2	d3			d1	d2	d3
25	rupa	0	0	1	1	0.4771	0.0000	0.0000	0.4771
26	selesai	1	0	0	1	0.4771	0.4771	0.0000	0.0000
27	sistem	0	1	1	2	0.1761	0.0000	0.1761	0.1761
28	spk	0	1	0	1	0.4771	0.0000	0.4771	0.0000
29	struktur	0	0	1	1	0.4771	0.0000	0.0000	0.4771
30	temu	0	0	1	1	0.4771	0.0000	0.0000	0.4771
31	timbang	0	1	0	1	0.4771	0.0000	0.4771	0.0000

Berdasarkan tabel 4.1 diatas dapat dijelaskan kata merupakan kata-kata yang telah di proses untuk membangun *index* dari sebuah dokumen, *TF (term frequency)* merupakan banyaknya kata yang muncul pada setiap dokumen, *DF (document frecuency)* merupakan kemunculan kata pada keseluruhan dokumen, *idf (inverse document frequency)* merupakan *log* (*tf*/*df)* seperti persamaan rumus 2.3. Dari tabel ini mendeskripsikan pengolahan kata sebelum dihitung kemiripanya menggunakan model ruang vector.

Tahapan Preproses *query*:

Setelah dilakukan pengindeksan terhadap koleksi dokumen, diinputkan query yang akan dilakukan pencocokan terhadap koleksi dokumen. *query* yang dihasilkan dari pembuatan *query* adalah “kritis disain sistem algoritma data temu basis”.

Tahapan – tahapan yang dilakukan

1. Menghilangkan tanda baca
   kritis disain sistem algoritma data temu basis
2. Mengubah istilah ke bentuk huruf kecil
   kritis disain sistem algoritma data temu basis
3. Menerapkan *stopword removal*
   kritis disain sistem algoritma data temu basis
4. Menerapkan stemming (mengembalikan kata ke kata dasar)
   kritis disain sistem algoritma data temu basis

5. Pembobotan, *query* yang telah di preproses dan sesuai dengan istilah hasil indexing pada koleksi dokumen disimpan ke dalam indexing *query*.

   Pada koleksi dokumen terdapat tiga dokumen, sehingga diperoleh N= 3 dan berdasarkan rumus 2.3 maka untuk istilah algoritma dimana istilah algoritma tersebut muncul pada pada 2 dokumen yaitu pada dokumen d_1 dan d_3 maka diperoleh df = 2, idf yang didapatkan adalah 0. 176.

   Dengan penerapan rumus yang sama maka idf setiap istilah selengkapnya dapat dilihat pada table 4.2

   Tabel 4.2 Hasil pembobotan index *query*

Kata	TF				Df	Idf	Bobot
	Q	d1	d2	d3			*query*
algoritma	1	1	0	1	2	0.1761	0.1761
basis	1	0	1	1	2	0.1761	0.1761
data	1	0	1	2	2	0.1761	0.1761
disain	1	0	0	1	1	0.4771	0.4771
kritis	1	0	0	1	1	0.4771	0.4771
sistem	1	0	1	1	2	0.1761	0.1761
temu	1	0	0	1	1	0.4771	0.4771

   Pada tabel 4.2 dilakukan proses pembobotan *query* terhadap kata kata yang ada pada *index* dokumen seperti yang telah dijelaskan pada tabel 4.1. pembobotan ini dilakukan untuk menghitung panjang vektor *query*.

Tahapan perhitungan kemiripan *query* terhadap dokumen dengan model ruang vektor berdaasarkan persamaan 2.7:

$$R(Q,d1) = \frac{(0.1761x0.1761) + (0.1761x0) + (0.1761x0) + (0.1761x0) + (0.1761x0) + (0.1761x0) + (0.1761x0)}{\sqrt{1.852}\ x\ \sqrt{0.806}}$$

$$R(Q,d1) = \frac{0.031 + 0 + 0 + 0 + 0 + 0 + 0}{1.36\ x\ 0.8983}$$

$$R(Q,d1) = \frac{0.031}{1.22}$$

$$R(Q,d1) = 0.025$$

$$R(Q,d2) = \frac{\begin{matrix}(0.1761x0)+(0.1761x0.1761)+(0.1761x0.1761)\\ +(0.4771x0)+(0.4771x0)+(0.1761x0.1761)\\ +(0.4771x0)\end{matrix}}{\sqrt{3.052}\ x\ \sqrt{0.806}}$$

$$R(Q,d2) = \frac{0+0.031+0.031+0+0+0.031+0}{1.747\ x\ 0.8983}$$

$$R(Q,d2) = \frac{0.093}{1.56}$$

$$R(Q,d2) = 0.059$$

$$R(Q,d3) = \frac{\begin{matrix}(0.1761x0.1761)+(0.1761x0.1761)+(0.1761x0.3522)\\ +(0.4771x0.4771)+(0.4771x0.4771)+(0.1761x0.1761)\\ +(0.4771x0.4771)\end{matrix}}{\sqrt{2.265}\ x\ \sqrt{0.806}}$$

$$R(Q,d3) = \frac{0.031+0.031+0.062+0.2276+0.2276+0.031+0.2276}{1.505\ x\ 0.8983}$$

$$R(Q,d3) = \frac{0.838}{1.352}$$

$$R(Q,d3) = 0.619$$

Dari hasil penghitungan relevansi pada langkah sebelumnya, maka koleksi dokumen tersebut dapat diurutkan dari yang paling relevan (diurut menurun) sebagai berikut:

Query yang dimasukkan “kritis disain sistem algoritma data temu basis”. Koleksi dokumen yang ditampilkan adalah D_3, D_2, D_1

Pendeteksian Kemiripan Dokumen Menggunakan algoritma *biword winnowing* dan *Jaccard Coefficient*

Pada tahapan ini akan dilakukan sebanyak jumlah dokumen yang berhasil dikembalikan oleh mesin pencari menggunakan model ruang vektor, pendeteksian akan dilakukan sesuai dengan urutan dokumen yang dihasil kan oleh mesin pencari yaitu: D3, D2 dan D1 terhadap dokumen uji pada awal pembuatan *query*.

Tahapan-tahapan yang dilakukan adalah:

1. *Whitespace Intensitivity* atau *preprocessing*, yaitu menghilangkan karakter yang tidak relevan seperti menghilangkan tanda baca dan mengubah huruf besar menjadi kecil. Sehingga terbentuk kalimat:

Dokumen Uji:

algoritma genetika yang memiliki kehandalan dalam menghasilkan output yang optimal dapat dimanfaatkan untuk menyelesaikan masalah tersebut dengan bantuan aplikasi pemilihan struktur data dan algoritma merupakan permasalahan yang kritis dalam disain sistem yang memungkinkan temu kembali dengan basis data berukuran besar secara efektif dan efisien

Dokumen 3:

pemilihan struktur data dan algoritma merupakan permasalahan yang kritis dalam disain sistem yang memungkinkan temu kembali dengan basis data berukuran besar secara efektif dan efisien

2. Proses selanjutnya adalah tokenisasi, yaitu pemotongan kata berbentuk *biword.* Sehingga terbentuk token kata *biword* seperti pada tabel 4.3.

Tabel 4.3 Hasil token *biword*

Token Dokumen Uji	Token Dokumen 3
[0] => algoritma genetika	[0] => pemilihan struktur
[1] => genetika yang	[1] => struktur data
[2] => yang memiliki	[2] => data dan
[3] => memiliki kehandalan	[3] => dan algoritma
[4] => kehandalan dalam	[4] => algoritma merupakan
[5] => dalam menghasilkan	[5] => merupakan permasalahan
[6] => menghasilkan output	[6] => permasalahan yang
[7] => output yang	[7] => yang kritis
[8] => yang optimal	[8] => kritis dalam
[9] => optimal dapat	[9] => dalam disain
[10] => dapat dimanfaatkan	[10] => disain sistem
[11] => dimanfaatkan untuk	[11] => sistem yang
[12] => untuk menyelesaikan	[12] => yang memungkinkan
[13] => menyelesaikan masalah	[13] => memungkinkan temu
[14] => masalah tersebut	[14] => temu kembali
[15] => tersebut dengan	[15] => kembali dengan

Tabel 4.3 Hasil token *biword* (lanjutan)

Token Dokumen Uji	Token Dokumen 3
[16] => dengan bantuan	[16] => dengan basis
[17] => bantuan aplikasi	[17] => basis data
[18] => aplikasi pemilihan	[18] => data berukuran
[19] => pemilihan struktur	[19] => berukuran besar
[20] => struktur data	[20] => besar secara
[21] => data dan	[21] => secara efektif
[22] => dan algoritma	[22] => efektif dan
[23] => algoritma merupakan	[23] => dan efisien
[24] => merupakan permasalahan	
[25] => permasalahan yang	
[26] => yang kritis	
[27] => kritis dalam	
[28] => dalam disain	
[29] => disain sistem	
[30] => sistem yang	
[31] => yang memungkinkan	
[32] => memungkinkan temu	
[33] => temu kembali	
[34] => kembali dengan	
[35] => dengan basis	
[36] => basis data	
[37] => data berukuran	
[38] => berukuran besar	
[39] => besar secara	
[40] => secara efektif	
[41] => efektif dan	
[42] => dan efisien	

3. Mengubah token *biword* menjadi nilai *MD5*.

Untuk mengubah sebuah token *biword* menjadi *MD5*, dapat dilakukan dengan fungsi *MD5* yang terdapat dalam bahasa pemrograman *PHP*. Setelah masing-masing token *biword* diubah menjadi *MD5*, akan diperoleh hasil sebagai berikut:

Dokumen Uji:

```
[0] => 998d2e98c5e38b03a951ee3294c0e553
[1] => 987bf6950cb20c2a1fb2a1e013b814e0
[2] => 83d01da1003236447620e49d25ca7b06
[3] => a531c2469c1103430a1dc95fd7a27e55
[4] => f0ce1a317c8eafa0d7bdbc3830a4ff67
[5] => 967676713e5d0114f3423feaff39acd2
[6] => 2db424971a31301d889fdc26fa25ba65
[7] => c1d893c885690cb45a600abe8d51e0d1
[8] => 625a209fb69c626bbcdbc68991a65137
[9] => 7415643910fb03611d5dac24fcb827e7
[10] => 3828e97b53b817f9d6e47b2610d22376
[11] => 580901301e97f174e26d0488853a2b27
[12] => 5a48a726ad11ea4612b1c43b133d8ab7
[13] => 403a826a8c033176054485f79c80594f
[14] => 573817939946a174c00322f36a8b96d2
[15] => 8f0c0e5ef42449c2a7a54baa46732cad
[16] => 7df7264f35cb214094281492de40e96a
[17] => eb1ed9b3d06370b90aaea8e7c36ffcaf
[18] => 0e8f3083477917e1a300a992f8c71e1e
[19] => 666853446c0a48fe21e1d7fb69c6be57
[20] => ed5435491272fb9b1d06c37556768b92
[21] => bdc7c27d0ea015ba2852311cde4d597b
```

Dokumen Uji (lanjutan):

```
[22] => f589496c0f5cd4fedf98c150c9f41025
[23] => a2ed243f4090db2f974af40e96b24f24
[24] => 3fbaecb1cede43a9d5802c21e46c053e
[25] => 5c706d3e8c99d2b2ff12a7e7c1af1ed0
[26] => f58a9cb63576c0366fd3a8c0d1966b8e
[27] => b457e902ea3609249fc993037896afe0
[28] => 525bf4a36d99e628db8a1ee78ff4d82c
[29] => 574e7cd71f01a8975b778cd7854e008b
[30] => 53521c2dcbbc4afa378c46b263cba6ef
[31] => 11edddb40dcb570e3f72bcea9583be6a
[32] => d5205afc6f66e43499b4c0c02a893b6d
[33] => 00787c2f6d67ed65f447741795c04c1d
[34] => de81d950e832b9fb26da258f634cd529
[35] => 4df0a258fe93fe7dc23a2fc264d3dd92
[36] => 9d43d6ea63c5b521c0bc19d1da2921be
[37] => 76885860e263060adb2c748f60d612b3
[38] => 1abb9c6c4e8b1c002f8a68cad72a94b4
[39] => d76818c8a52c915e89f78ee600b59fcf
[40] => cb1e4a6819a6cbad921060a06b0b0eae
[41] => 8f19637469846759bde2ff2d169cfd4f
[42] => 7419728e91e3df64274c5a663d3381fa
```

Dokumen 3:

```
[0] => 666853446c0a48fe21e1d7fb69c6be57
[1] => ed5435491272fb9b1d06c37556768b92
[2] => bdc7c27d0ea015ba2852311cde4d597b
[3] => f589496c0f5cd4fedf98c150c9f41025
[4] => a2ed243f4090db2f974af40e96b24f24
[5] => 3fbaecb1cede43a9d5802c21e46c053e
```

```
[6] => 5c706d3e8c99d2b2ff12a7e7c1af1ed0
[7] => f58a9cb63576c0366fd3a8c0d1966b8e
[8] => b457e902ea3609249fc993037896afe0
[9] => 525bf4a36d99e628db8a1ee78ff4d82c
[10] => 574e7cd71f01a8975b778cd7854e008b
[11] => 53521c2dcbbc4afa378c46b263cba6ef
[12] => 11edddb40dcb570e3f72bcea9583be6a
[13] => d5205afc6f66e43499b4c0c02a893b6d
[14] => 00787c2f6d67ed65f447741795c04c1d
[15] => de81d950e832b9fb26da258f634cd529
[16] => 4df0a258fe93fe7dc23a2fc264d3dd92
[17] => 9d43d6ea63c5b521c0bc19d1da2921be
[18] => 76885860e263060adb2c748f60d612b3
[19] => 1abb9c6c4e8b1c002f8a68cad72a94b4
[20] => d76818c8a52c915e89f78ee600b59fcf
[21] => cb1e4a6819a6cbad921060a06b0b0eae
[22] => 8f19637469846759bde2ff2d169cfd4f
[23] => 7419728e91e3df64274c5a663d3381fa
```

Setelah didapatkan nilai MD5 masing-masing token *biword* yang dibentuk, selanjutnya akan dihitung nilai *hash* menggunakan persamaan *Rolling Hash.* Nilai-nilai *hash* ini akan dipilih nantinya untuk dijadikan *fingerprint.*

Berikut adalah tabel 4.4 yang merupakan hasil perhitungan nilai *hash* masing-masing token *biword*:

Tabel 4.4. Nilai *hash* token *biword*

Token Dokumen Uji	Token Dokumen 3
[0] => 258247033209 [1] => 259631046226 [2] => 259774433926 [3] => 326142312171 [4] => 367663887435 [5] => 239205881598 [6] => 295388782597 [7] => 350204784785 [8] => 239032808521 [9] => 228867238637 [10] => 233383424288 [11] => 228358173027 [12] => 281692812831 [13] => 232031181210 [14] => 229283220218 [15] => 300115964114 [16] => 309814531481 [17] => 398201776708 [18] => 282676831411 [19] => 232259278917 [20] => 380671597868 [21] => 407050206300 [22] => 336732088481 [23] => 357586293296 [24] => 323636969615 [25] => 281046219092 [26] => 351444753101 [27] => 330809299098 [28] => 244689925959 [29] => 247277474850 [30] => 228656225616 [31] => 264392933473 [32] => 331223223744 [33] => 217731976722 [34] => 387576660737 [35] => 307332697348 [36] => 294460894349 [37] => 235961288743 [38] => 307212293792 [39] => 333728840732 [40] => 388497130971 [41] => 285700163806 [42] => 230906887617	[0] => 232259278917 [1] => 380671597868 [2] => 407050206300 [3] => 336732088481 [4] => 357586293296 [5] => 323636969615 [6] => 281046219092 [7] => 351444753101 [8] => 330809299098 [9] => 244689925959 [10] => 247277474850 [11] => 228656225616 [12] => 264392933473 [13] => 331223223744 [14] => 217731976722 [15] => 387576660737 [16] => 307332697348 [17] => 294460894349 [18] => 235961288743 [19] => 307212293792 [20] => 333728840732 [21] => 388497130971 [22] => 285700163806 [23] => 230906887617

Nilai perhitungan pada tabel 4.4 didapatkan berdasarkan persamaan 2.14 berdasarkan nilai enkripsi *md5* pada tahapan sebelumya.

4. Pembentukan *window* dari nilai *hash* yang telah diperoleh.

Misalkan ukuran *window w* yang digunakan adalah 4, maka diperoleh hasil pembagian token *hash* sebagai berikut:

Kalimat 1:

[**258247033209** 259631046226 259774433926 326142312171]

[**259631046226** 259774433926 326142312171 367663887435]

[259774433926 326142312171 367663887435 **239205881598**]

[326142312171 367663887435 239205881598 295388782597]

[367663887435 239205881598 295388782597 350204784785]

[239205881598 295388782597 350204784785 **239032808521**]

[295388782597 350204784785 239032808521 **228867238637**]

[350204784785 239032808521 228867238637 233383424288]

[239032808521 228867238637 233383424288 **228358173027**]

[228867238637 233383424288 228358173027 281692812831]

[233383424288 228358173027 281692812831 232031181210]

[228358173027 281692812831 232031181210 229283220218]

[281692812831 232031181210 **229283220218** 300115964114]

[232031181210 229283220218 300115964114 309814531481]

[229283220218 300115964114 309814531481 398201776708]

[300115964114 309814531481 398201776708 **282676831411**]

[309814531481 398201776708 282676831411 **232259278917**]

[398201776708 282676831411 232259278917 380671597868]

[282676831411 232259278917 380671597868 407050206300]

[232259278917 380671597868 407050206300 336732088481]

[380671597868 407050206300 **336732088481** 357586293296]

[407050206300 336732088481 357586293296 **323636969615**]

[336732088481 357586293296 323636969615 **281046219092**]

[357586293296 323636969615 281046219092 351444753101]

[323636969615 281046219092 351444753101 330809299098]

[281046219092 351444753101 330809299098 **244689925959**]

[351444753101 330809299098 244689925959 247277474850]

[330809299098 244689925959 247277474850 **228656225616**]

[244689925959 247277474850 228656225616 264392933473]

[247277474850 228656225616 264392933473 331223223744]

[228656225616 264392933473 331223223744 **217731976722**]

[264392933473 331223223744 217731976722 387576660737]

[331223223744 217731976722 387576660737 307332697348]

[217731976722 387576660737 307332697348 294460894349]

[387576660737 307332697348 294460894349 **235961288743**]

[307332697348 294460894349 235961288743 307212293792]

[294460894349 235961288743 307212293792 333728840732]

[235961288743 307212293792 333728840732 388497130971]

[307212293792 333728840732 388497130971 **285700163806**]

[333728840732 388497130971 285700163806 **230906887617**]

Dokumen 3:

[**232259278917** 380671597868 407050206300 336732088481]
[380671597868 407050206300 **336732088481** 357586293296]
[407050206300 336732088481 357586293296 **323636969615**]
[336732088481 357586293296 323636969615 **281046219092**]
[357586293296 323636969615 281046219092 351444753101]
[323636969615 281046219092 351444753101 330809299098]
[281046219092 351444753101 330809299098 **244689925959**]
[351444753101 330809299098 244689925959 247277474850]

[330809299098 244689925959 247277474850 **228656225616**]
[244689925959 247277474850 228656225616 264392933473]
[247277474850 228656225616 264392933473 331223223744]
[228656225616 264392933473 331223223744 **217731976722**]
[264392933473 331223223744 217731976722 387576660737]
[331223223744 217731976722 387576660737 307332697348]
[217731976722 387576660737 307332697348 294460894349]
[387576660737 307332697348 294460894349 **235961288743**]
[307332697348 294460894349 235961288743 307212293792]
[294460894349 235961288743 307212293792 333728840732]
[235961288743 307212293792 333728840732 388497130971]
[307212293792 333728840732 388497130971 **285700163806**]
[333728840732 388497130971 285700163806 **230906887617**]

Nilai *hash* yang dicetak tebal adalah nilai *hash* terkecil yang dipilih pada setiap *window* untuk menjadi *fingerprint* dokumen.

Maka diperoleh nilai *hash* minimum masing-masing dokumen:

Dokumen Uji:

[258247033209,0] [259631046226,1] [239205881598,5]
[239032808521,8] [228867238637,9] [228358173027,11]
[229283220218,14] [282676831411,18] [232259278917,19]
[336732088481,22] [323636969615,24] [281046219092,25]
[244689925959,28] [228656225616,30] [217731976722,33]
[235961288743,37] [285700163806,41] [230906887617,42]

Dokumen 3:

[232259278917,0][336732088481,3] [323636969615,5]
[281046219092,6][244689925959,9][228656225616,11]
[217731976722,14][235961288743,18][285700163806,22]
[230906887617,23]

Nilai-nilai *hash* minimum (*fingerprint)* yang diperoleh berdasarkan posisi indeks nya, jika diubah kembali menjadi token *biword* akan terlihat frasa mana yang memiliki *fingerprint* yang sama antara kedua kalimat yang diuji.

Berikut ini adalah *biword* yang dianggap memiliki nilai *fingerprint* yang sama.

Tabel 4.5 Token *biword* dengan *fingerprint* yang sama

Dokumen Uji	Dokumen 3
[19] pemilihan struktur	[0] pemilihan struktur
[22] dan algoritma	[3] dan algoritma
[24] merupakan permasalahan	[5] merupakan permasalahan
[25] permasalahan yang	[6] permasalahan yang
[28] dalam disain	[9] dalam disain
[30] sistem yang	[11] sistem yang
[33] temu kembali	[14] temu kembali
[37] data berukuran	[18] data berukuran
[41] efektif dan	[22] efektif dan
[42] dan efisien	[23] dan efisien

Pada tabel 4.5 menjelaskan *biword* yang sama antar kedua dokumen, nilai didepan kata menyatakan urutan *biword* yang terbentuk.

5. Proses selanjutnya adalah menghuitung *similarity.* Perhitungan *similarity* dapat dilakukan dari hasil pemilihan nilai *fingerprint* setiap dokumen .menggunakan persamaan 2.12

   Similaritas(duji,d3) = $\frac{|dok\ Uji(di) \cap dok\ 3(dj)|}{|dok\ Uji(di) \cup dok\ 3(dj)|}\ x\ 100\% = \frac{10}{18} = 55.5\%$

Lakukan langkah yang sama antara dokumen uji dan dokumen 2, Antara dokumen uji dan dokumen 1, hasinya sebagai berikut:

Similaritas(duji,d2) = $\frac{|dok\ Uji(di) \cap dok\ 2(dj)|}{|dok\ Uji(di) \cup dok\ 2(dj)|}\ x\ 100\% = \frac{0}{23} = 0\%$

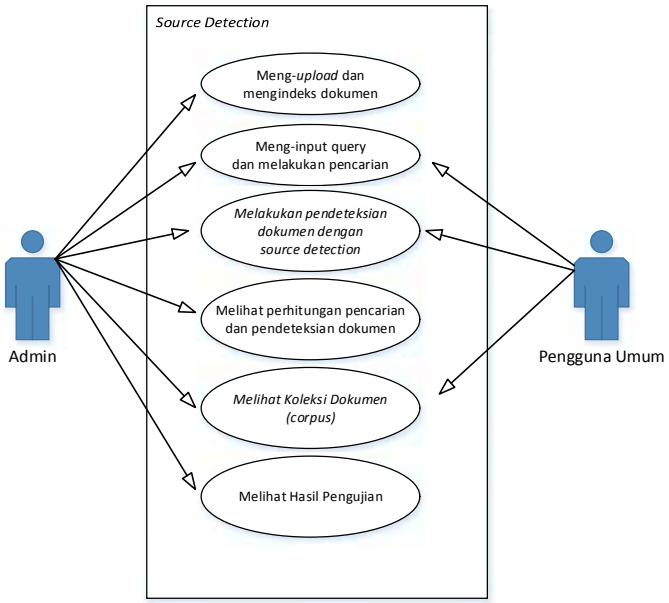
Similaritas(duji,d1) = $\frac{|dok\ Uji(di) \cap dok\ 1(dj)|}{|dok\ Uji(di) \cup dok\ 1(dj)|}\ x\ 100\% = \frac{7}{18} = 38.88\%$

4.5 Perancangan Aplikasi

Pada tahap ini akan dibahas tentang perancangan aplikasi *source detection* berdasarkan tahapan analisa yang telah dilakukan sebelumnya. Adapun perancangan yang akan dibuat adalah *use case diagram*, perancangan basis data, perancangan struktur menu dan perancangan *interface.*

4.5.1 *Use Case Diagram*

Use case diagram menjelaskan siapa-siapa saja actor yang terlibat atau dapat menggunakan sistem *Source Detection* ini dan apa saja yang dapat dilakukan (scenario) oleh actor terhadap sistem. *Actor* adalah semua yang berhubungan langsung ke sistem, yang memberi input atau menerima informasi dari sistem. *Use case diagram* untuk sistem *source detection* yang akan dibangun dapat dilihat pada Gambar 4.22.



Gambar 4.22 *Use Case Diagram source detection*

Berdasarkan gambar 4.22 terdapat dua *actor* yang terlibat dalam sistem ini, yaitu admin dan pengguna umum. Ketika login ke sistem, admin akan diminta menginputkan user id dan password. Admin dapat melakukan enam skenario yaitu, meng-*upload* dokumen dan mengindeks dokumen di dalam *corpus*, menginputkan query dan menjalankan proses pencarian, melakukan pendeteksian

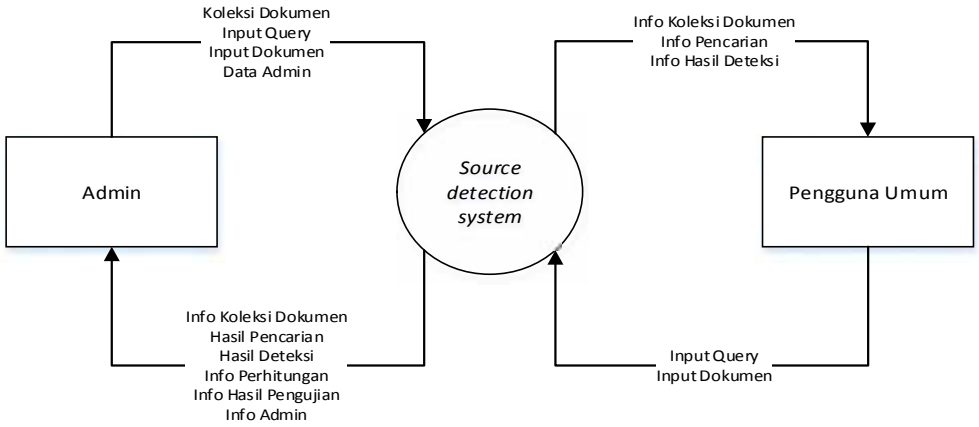
dokumen, admin juga dapat menampilkan nilai relevansi yang diperoleh dari hasil perhitungan sistem pencarian dan pendeteksian dokumen, dapat melihat seluruh dokumen yang ada di dalam corpus dan dapat melihat hasil pengujian. Sedangkan pengguna umum hanya dapat melakukan tiga skenario yaitu, menginputkan query dan menjalankan proses pencarian, melakukan pendeteksian dokumen dan melihat seluruh dokumen yang ada di dalam *corpu*s.

4.5.2 Deskripsi Fungsional

Deskripsi fungsional adalah gambaran fungsional sistem secara umum yang akan dirancang. Deskripsi fungsional menggambarkan alur proses dan aliran data dari sebuah sistem, seperti *Context Diagram, DFD* dan *ERD*. Deskripsi fungsional ini akan menjadi dasar perancangan sebuah sistem

a. ***Context Diagram***

Context Diagram berfungsi untuk menggambarkan proses kerja sistem secara umum. *Context Diagram* merupakan *Data Flow Diagram* yang menggambarkan garis besar operasional sistem. *Context Diagram* terdiri dari entitas, proses tunggal dan *data flow*. Semua yang berinteraksi dengan sistem disebut dengan entitas, dan data *flow* adalah aliran data. Pada sistem *source detection* yang menjadi entitas pada *context diagram* adalah Admin dan Pengguna umum. Gambar 4.23 berikut adalah *context diagram* dari sistem *source detection* yang di buat.



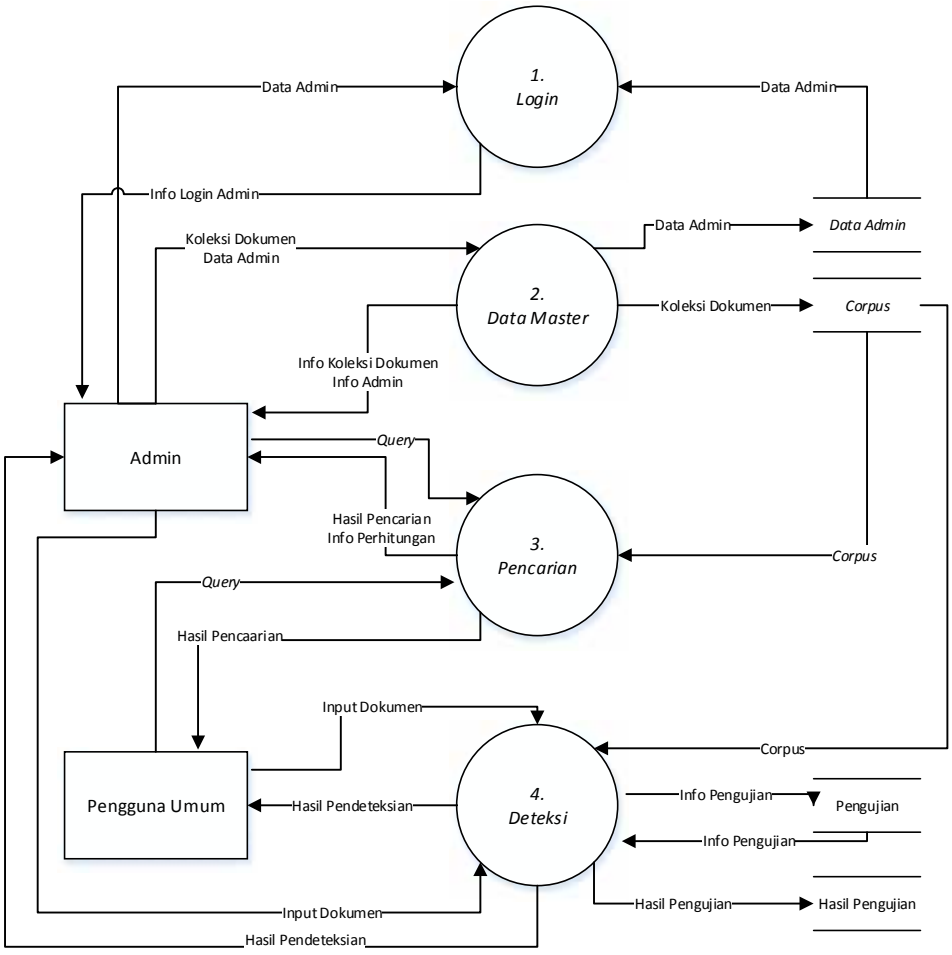
Gambar 4.23 *Context Diagram*

b. ***Data Flow Diagram (DFD)***

Data Flow Diagram (DFD) adalah diagram yang digunakan untuk menggambarkan aliran data dalam sebuah sistem, DFD sering digunakan untuk menggunakan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir, atau lingkungan fisik dimana data tersebut tersimpan.

DFD* Level 1 *Source Detection System

DFD Level 1 terdiri atas dua entitas dan tiga proses yang saling berhubungan. Proses-proses tersebut diantaranya pengelolaan koleksi dokumen, pencarian dan deteksi. Untuk lebih jelasnya entitas dan proses tersebut pada gambar 4.24.



Gambar 4.24 ***Data Flow Diagram (DFD) Level 1***

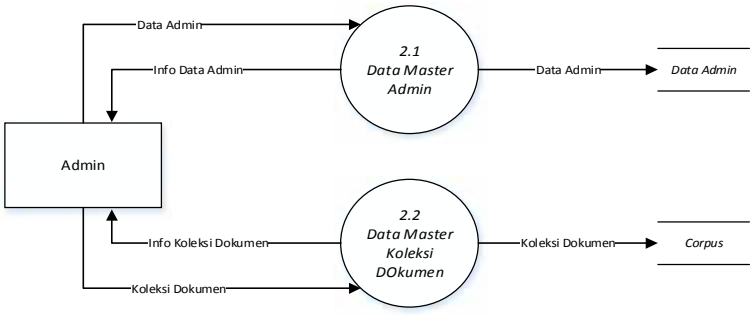
Berikut penjelasan dari gambar 4.24 akan di jelaskan pada tabel 4.6.

Tabel 4.6 Proses *DFD* Level 1 *source detection system*

No.	Nama Proses	Deskripsi
1	Login	Proses Login dilakukan untuk melakukan validasi kebenaran data login yang di *input* oleh admin berupa *username* dan *password*
2	Data Master	Proses Data Master dilakukan oleh admin yang bertujuan untuk mengorganisir koleksi dokumen ke sistem dan mengelola data admin itu sendiri
3	Pencarian	Proses Pencarian dapat dilakukan oleh admin maupun pengguna umum. Pencarian ini digunakan untuk mencari koleksi dokumen berdasarkan *query* yang di-*input* kan. Untuk admin, sistem akan memberikan informasi berupa perhitungan hasil pencarian.
4	Deteksi	Proses Deteksi dapat dilakukan oleh admin maupun pengguna umum. Proses deteksi ini digunakan untuk mendeteksi sumber dan mendeteksi kemiripan dokumen yang di-*input* kan ke dalam sistem terhadap koleksi dokumen yang telah tersimpan. Setiap proses deteksi, sistem akan menyimpan riwayat pendeteksian dan menampilkannya kepada admin. Pada proses deteksi ini juga admin akan memberikan informasi berupa perhitungan hasil pendeteksian.

DFD Level 2 Proses 2 Data Master

DFD Level 2 Proses 2 Data Master terdiri proses pengelolaan koleksi dokumen dan pengelolaan data admin. Untuk lebih jelasnya entitas dan proses tersebut pada gambar 4.25.



Gambar 4.25 *Data Flow Diagram (DFD)* Level 2 Proses Data Master

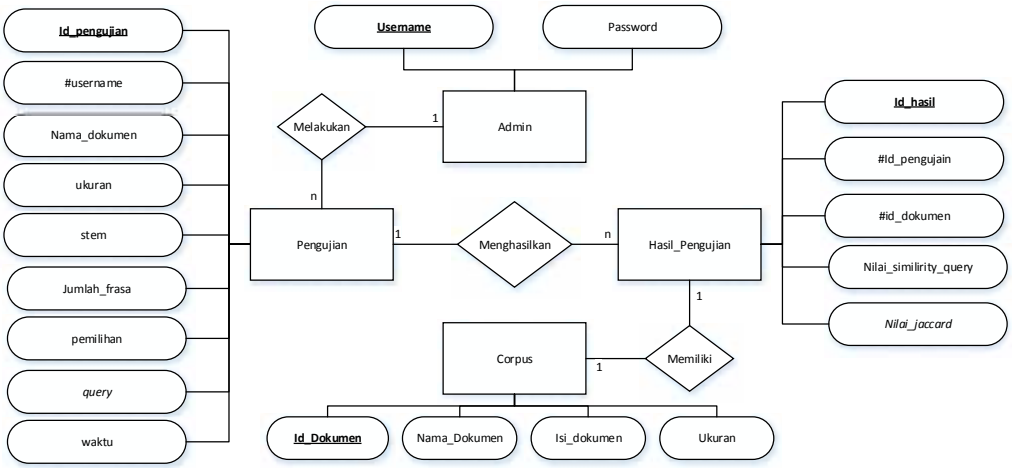
Berikut penjelasan dari gambar 4.25 akan di jelaskan pada tabel 4.7.

Tabel 4.7 Proses *DFD* Level 1 *source detection system*

No.	Nama Proses	Deskripsi
2.1	Data Master Admin	Proses Data Master Admin, dilakukan oleh admin untuk mengelola data admin.
2.2	Data Master Koleksi Dokumen	Proses Data Master Koleksi Dokumen dilakukan oleh admin yang bertujuan untuk mengorganisir koleksi dokumen ke sistem.

c. Entity Relationship Diagram (ERD)

ERD digunakan untuk menggambarkan tabel-tabel yang berelasi beserta semua atribut yang berada dalam tabel tersebut. Berikut adalah ERD dari *source detection system* pada gambar 4.26.



Gambar 4.26 *Entity Relationship Diagram (ERD)*

Berdasarkan rancangan *ERD* pada gambar 4.26, *ERD* ini akan menggambarkan data-data yang dibutuhkan dalam sistem *source detection* sesuai dengan *datastore* yang ada pada *DFD*. Penjelasan *ERD* dapat dilihat pada tabel 4.8 berikut:

Tabel 4.8 Keterangan Entitas Pada *ERD*

No	Nama	Deskripsi	Atribut	Primary key	Foreign Key
1	Admin	Menyimpan informasi yang digunakan admin untuk melakukan validasi kebenaran data ketika akan masuk (*login*) kedalam sistem	-username -password	-username	-
2	Corpus	Menyimpan informasi koleksi dokumen yang akan di proses dan digunakan pada sistem ini	-id_dokumen -nama_dokumen -isi_dokumen -ukuran	-id_dokumen	-
3	Pengujian	Menyimpan informasi setiap pengujian berupa konfigurasi yang digunakan saat pendeteksian	-id_pengujian -username -nama_dokumen -ukuran -stem -jumlah_frasa -pemilihan -*query* -waktu	Id_pengujian	username
4	Hasil_Pengujian	Menyimpan informasi hasil yang didapatkan pada saat proses pendeteksian berupa nama dokumen beserta nilai kemiripan	-Id_hasil -Id_pengujian -id_dokumen -nilai_similirity_query -nilai_jaccard	Id_hasil	Id_pengujian Id_dokumen

4.5.3 Perancangan *Database*

Perancangan *database* yang dibangun pada aplikasi bertujuan untuk menyimpan hasil proses *source detection*. *Conceptual* data model digunakan untuk mengetahui tipe-tipe data yang digunakan dalam database aplikasi *source detection*. Berikut adalah *Conceptual* data model yang dirancang pada aplikasi ini.

Tabel 4.9 *Conceptual* Data Model Tabel pengujian

Field	Type	Null
Id_pengujian	*Int (11)*	No
Nama_dokumen	*Varchar(50)*	No
Ukuran	*Int(10)*	No
Stem	*Varchar(3)*	No
Jumlah_frasa	*Int(10)*	No
Pemilihan	*Varchar(10)*	No
Query	*Text*	No
Waktu	*Decimal(20,4)*	No

Tabel 4.10 *Conceptual* Data Model Tabel hasil_pengujian

Field	Type	Null
Id_hasil	*Int (11)*	No
Id_pengujian	*Int(11)*	No
Nama	*Varchar(50)*	No
Ukuran	*Int(11)*	No
Nilai_similirity_query	*Decimal(10,2)*	No
Nilai_jaccard	*Decimal(10,2)*	No

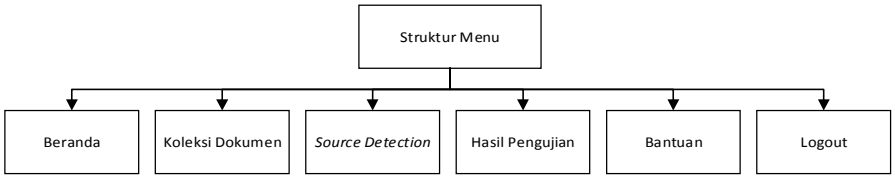
Pada tabel 4.9 dan 4.10 mendeskripsikan kebutuhan data dan informasi untuk membangun *database* yang akan digunakan pada sistem ini. *Field* merupakan nama atribut, *Type* merupakan jenis data yang dapat disimpan pada setiap *field* dan *null* merupakan diperbolehkan atau tidaknya suatu *field* itu kosong.

4.5.4 Perancangan *File* Teks

Dalam perancangan sistem tugas akhir ini, untuk proses penyimpanan data maupun informasi pada proses pencarian dan pendeteksian dokumen tidak menggunakan database relasional, melainkan *flat file* yang menggunakan *file* teks (*plain text*) sebagai media penyimpanannya. Penggunaan *flat* file bertujuan untuk meningkatkan kecepatan dalam proses pencarian dokumen. Sehingga seluruh informasi dari hasil proses tokenisasi, pengindeksan (*indexing*), pembobotan lokal (*tf*), dan perhitungan nilai relevansi akan disimpan dalam *file* teks. Daftar kata stop-words disimpan dalam *file* Stop.txt yang sebelumnya telah disimpan dalam pembangun sistem dan berjumlah 352 kata. Daftar stop-words ini berdasarkan sumber dari http://www.scribd.com/doc/61824071/DAFTAR-PUSTAKA. Kamus kata dasar Bahasa indonesia disimpan dalam *file* kamus-ind.txt yang sebelumnya telah disimpan dalam pembangun sistem dan berjumlah 28526 kata. Setelah sistem dijalankan, informasi dari hasil proses tokenisasi, pengindeksan dan pembobotan lokal (*tf*) akan disimpan ke dalam *file* Filelist.txt, Indexing.txt. *File* Filelist.txt akan menyimpan informasi berupa id dokumen, dan nama dokumen. Sedangkan file Indexing.txt menyimpan informasi dari hasil tokenisasi, pengindeksan, dan pembobotan lokal, yaitu berupa kata (*term*) dan id dokumen terkait dimana kata tersebut muncul beserta frekuensi kemunculannya (*tf*).

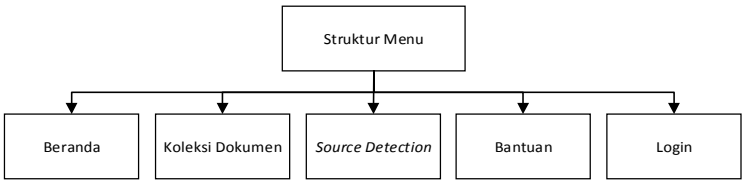
4.5.5 Perancangan Struktur Menu

Perancangan struktur menu adalah tahap merancang menu-menu yang dapat digunakan pengguna untuk menjalankan aplikasi, sehingga dapat memudahkan pengguna dalam memilih proses yang akan dijalankannya. Untuk lebih jelasnya dapat dilihat pada gambar strktur menu berikut:



Gambar 4.27 Rancangan struktur menu admin

Struktur menu pada gambar 4.27 merupakan struktur menu yang ada pada admin, yang terdiri dari menu beranda, menu koleksi dokumen, menu *source detection*, menu hasil pengujian, menu bantuan dan menu logout.

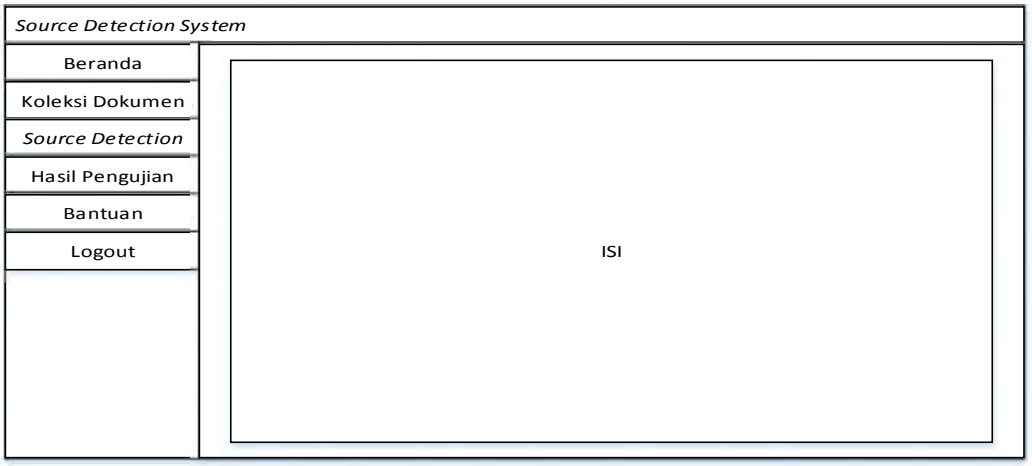


Gambar 4.28 Rancangan struktur menu pengguna umum

Struktur menu pada gambar 4.28 merupakan struktur menu yang ada pada pengguna umum, yang terdiri dari menu beranda, menu koleksi dokumen, menu *source detection*, menu bantuan dan menu login.

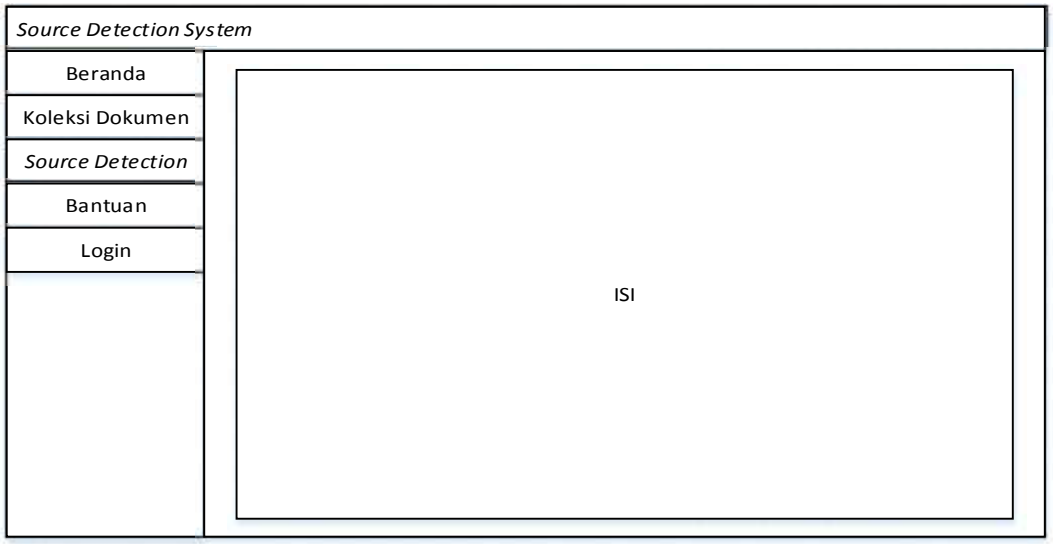
4.5.6 Perancangan *Interface*

Interface adalah sarana pengembangan sistem yang digunakan untuk membuat komunikasi lebih mudah, konsisten antara aplikasi dengan pemakaiannya. Penekanan *interface* meliputi tampilan yang baik dan mudah dipahami. Berikut adalah rancangan *interface* yang akan dibangun:



Gambar 4.29 Rancangan *interface* admin

Rancangan *interface* pada gambar 4.29 merupakan tampilan yang digunakan jika *actor* sebagai admin, dengan menu sesuai dengan rancangan struktur menu pada gambar 4.27.



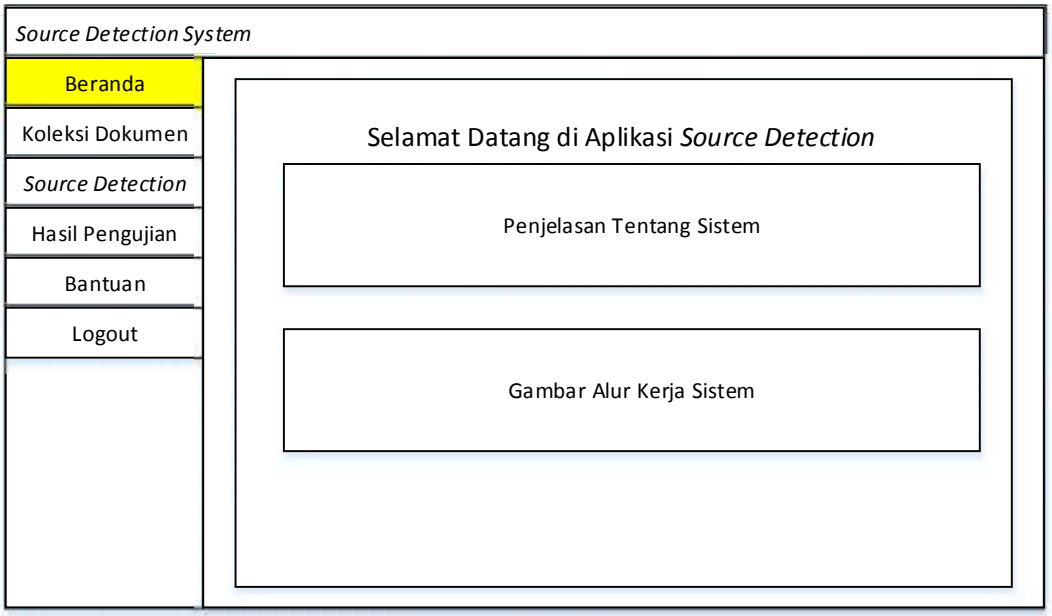
Gambar 4.30 Rancangan *interface* pengguna umum

Rancangan *interface* pada gambar 4.30 merupakan tampilan yang digunakan jika *actor* sebagai pengguna umum, dengan menu sesuai dengan rancangan struktur menu pada gambar 4.28.

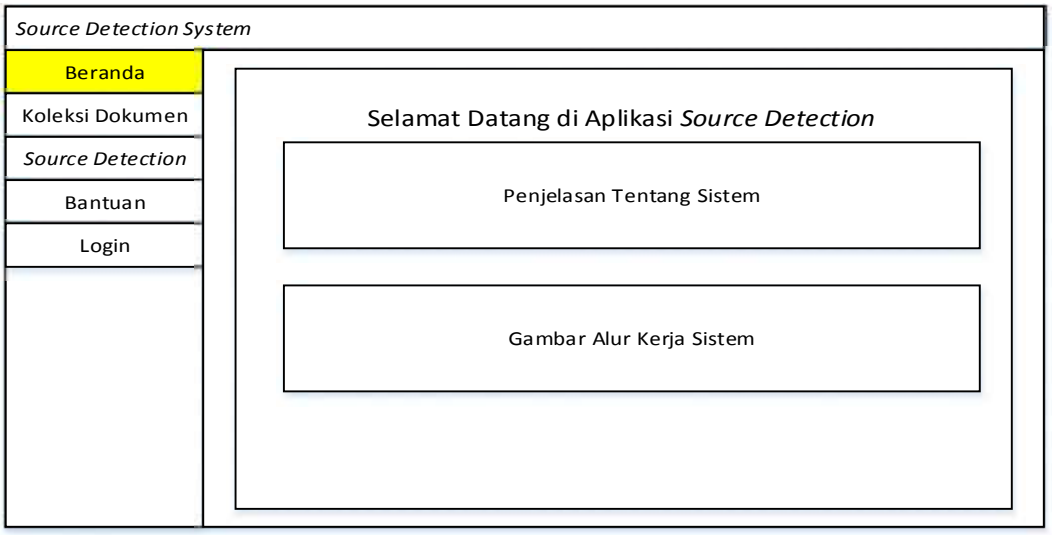
4.5.6.1 Rancangan menu Halaman utama (beranda)

Menu beranda adalah halaman utama yang ditampilkan aplikasi kepada pengguna. Halaman ini berisi tentang definisi atau sekilas penjelasan tentang sistem dan gambar alur kerja sistem. Halaman ini juga mendeskripsikan tujuan dari aplikasi ini.

Berikut adalah rancangan menu halaman utama pada *actor* admin yang digambarkan pada gambar 4.31, pada halaman ini hanya berupa informasi singkat mengenai sistem *source detection.*



Gambar 4.31 *Interface* Halaman utama admin

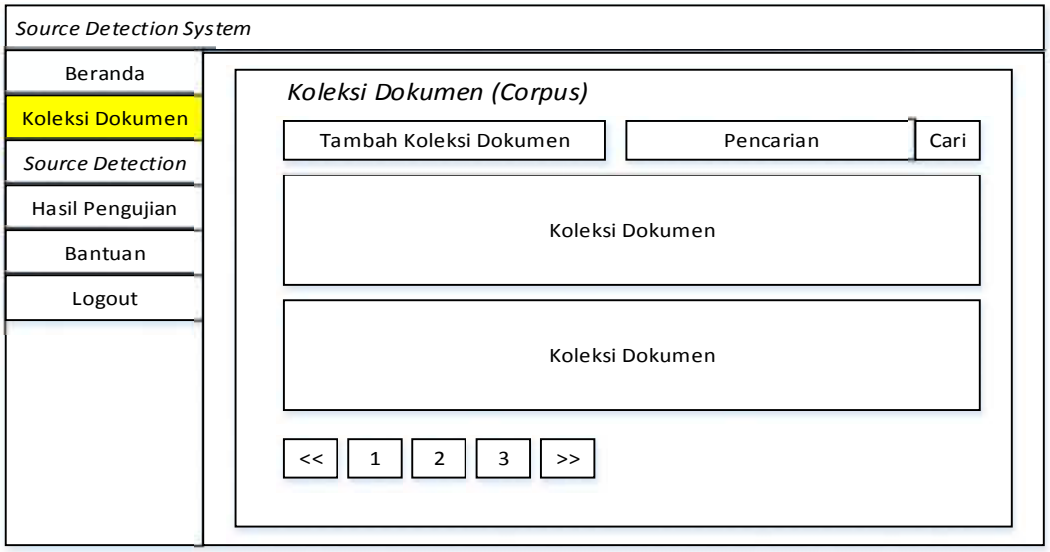


Gambar 4.32 *Interface* Halaman utama pengguna umum

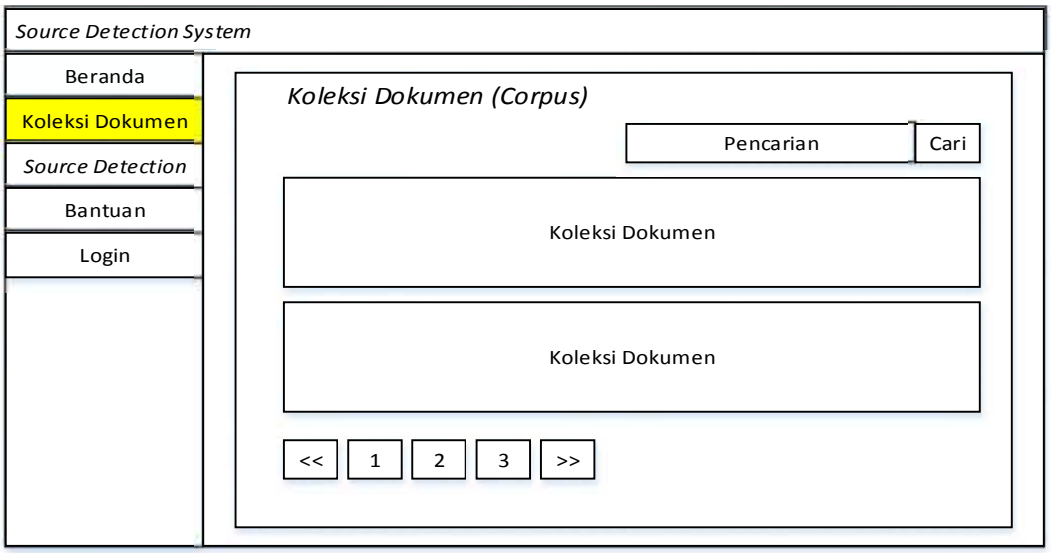
Rancangan *interface* menu halaman utama pada gambar 4.32 merupakan rancangan halaman jika *actor* sebagai pengguna umum, halaman ini sama halnya dengan rancangan pada gambar 4.31 namun pada rancangan ini menyesuaikan dengan struktur menu pada pengguna umum.

4.5.6.2 Rancangan menu Halaman Koleksi Dokumen

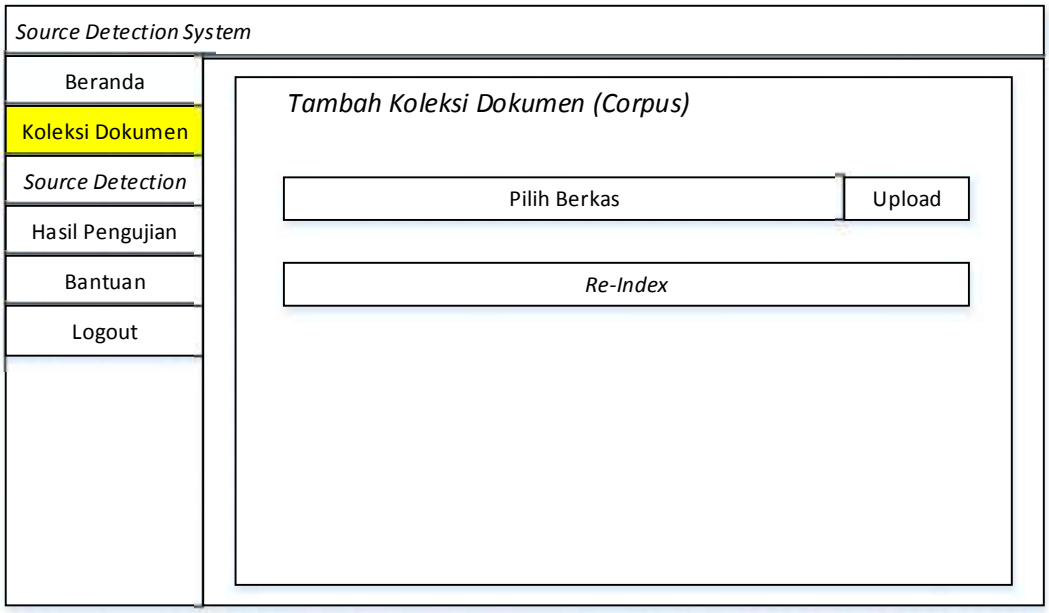
Menu Koleksi Dokumen adalah halaman yang menampilkan keseluruhan koleksi dokumen yang ada pada sistem, pada menu ini untuk admin dapat melakukan penambahan koleksi dokumen, mengindeks dokumen dan melakukan pencarian seperti yang tergambar pada gambar 4.33. Sedangkan untuk pengguna umum hanya dapat melakukan pencarian seperti yang tergambar pada gambar pada 4.34.



Gambar 4.33 *Interface* Halaman Koleksi Dokumen admin



Gambar 4.34 *Interface* Halaman Koleksi Dokumen pengguna umum

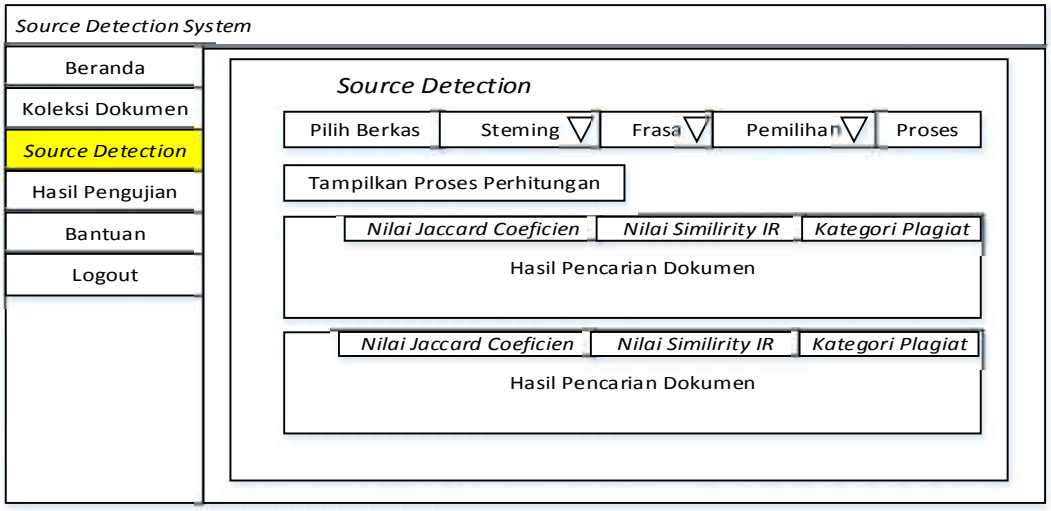


Gambar 4.35 *Interface* Halaman Tambah Koleksi Dokumen

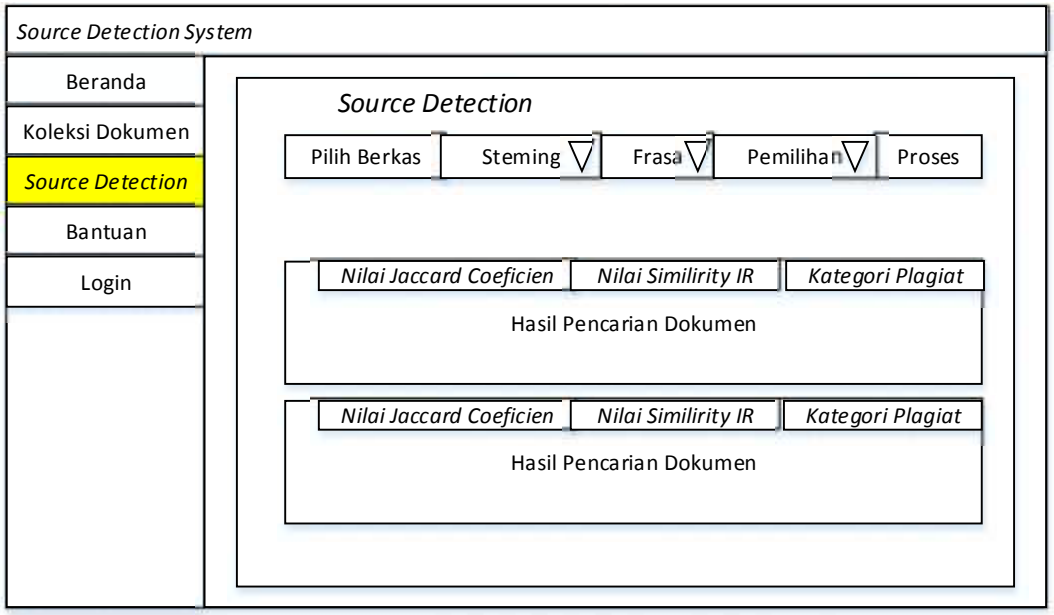
Rancangan *interface* pada gambar 4.35 adalah rancangan untuk *actor* sebagai admin, halaman ini akan muncul ketika admin melakukan klik tambah koleksi dokumen pada gambar 4.33. Halaman ini berfungsi untuk menambahkan koleksi dokumen serta melakukan proses pengindeksan kembali terhadap koleksi dokumen.

4.5.6.3 Rancangan menu Halaman *Source Detection*

Menu *Source Detection* adalah halaman yang merupakan inti dari tugas akhir ini, pada halaman ini dapat melakukan pendeteksian sumber dokumen yang di-*input* terhadap koleksi dokumen pada sistem. Halaman yang ditampilkan pada admin dan pengguna umum hampir sama, namun pada admin terdapat penjelasan perhitungan sedangnkan pengguna umum tidak. Pada gambar 4.36 rancangan *interface* menu *source detection* pada *actor* admin terdapat proses perhitungan yang dapat dilihat oleh admin, informasi lainnya berupa nama dokumen yang berhasil dideteksi, nilai *cosine similarity*, nilai *Jaccard Coefficient* serta kategori plagiat dari dokumen. Pada gambar 4.37 rancangan *interface* menu *source detection* untuk *actor* pengguna umum memiliki informasi yang sama seperi gambar 4.36, hanya saja pada rancangan ini tidak ditampilakan proses perhitungan.

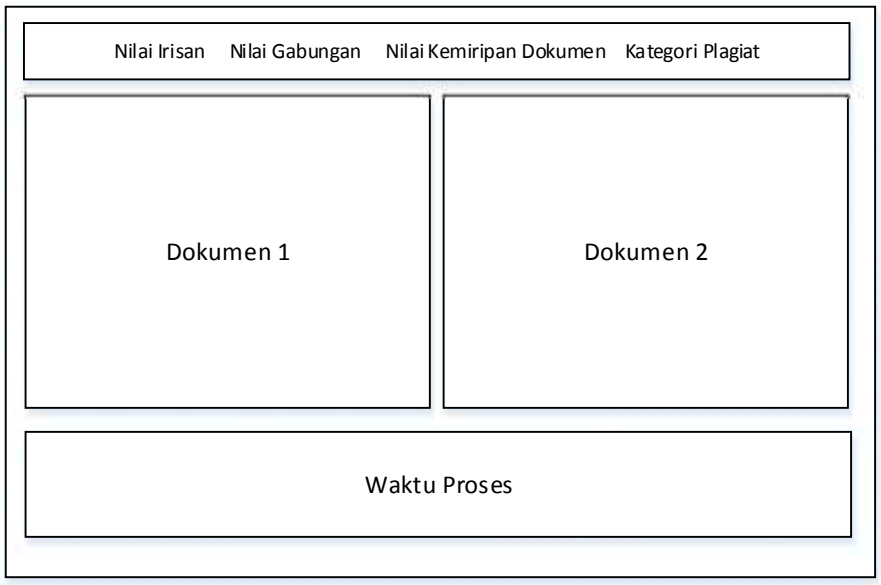


Gambar 4.36 *Interface* Halaman Hasil *Source Detection* admin



Gambar 4.37 *Interface* Halaman Hasil *Source Detection* pengguna umum

Jika Nilai *Jaccard Coefficient* di-klik maka akan muncul rincian dari dokumen uji dan dokumen hasil pencarian seperti gambar 4.38 berikut:

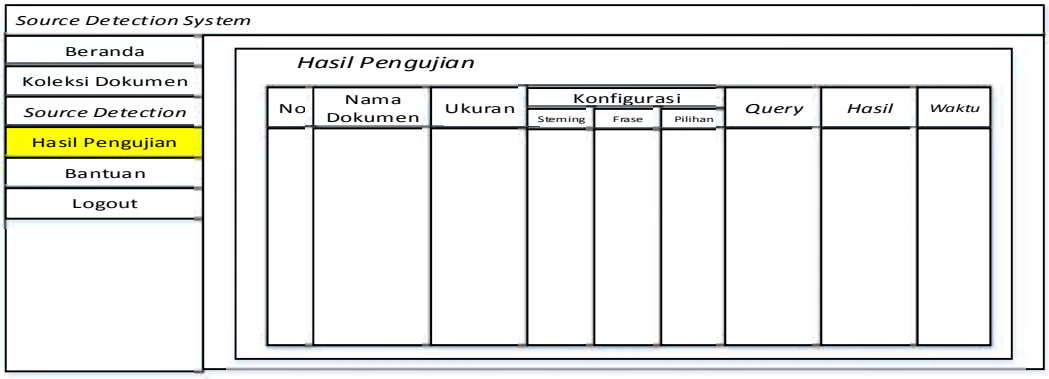


Gambar 4.38 *Interface* Halaman Hasil *biword winnowing*

Berdasarkan gambar 4.38 rancangan *interface* hasil *biword winnowing* memberikan informasi persamaan jumlah kata yang sama, nilai kemiripan antar dokumen, kategori plagiat serta cuplikan teks yang dianggap sama antar kedua dokumen.

4.5.6.4 Rancangan menu Halaman Hasil Pengujian

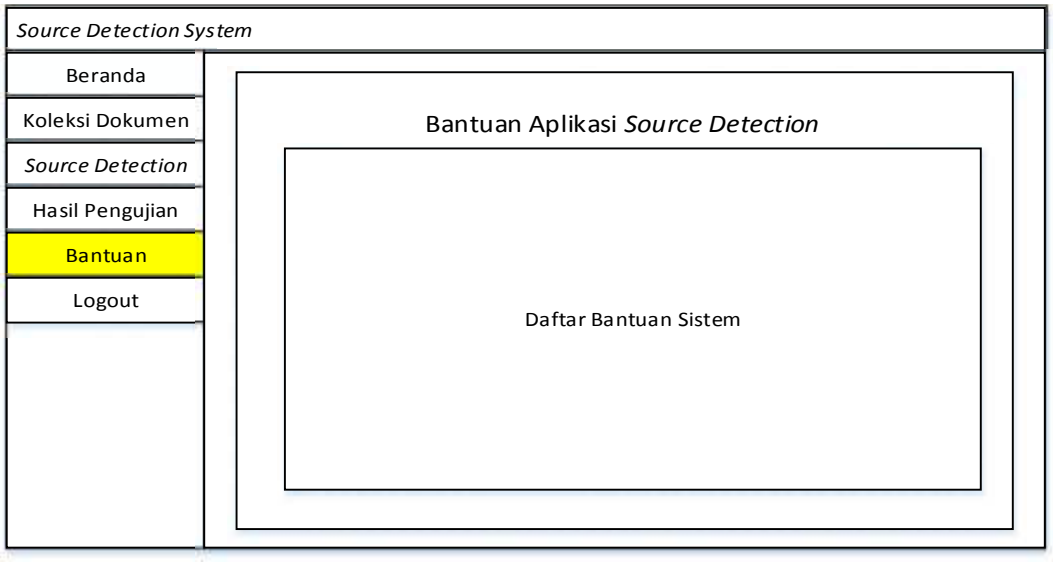
Menu Hasil Pengujian adalah halaman yang menampilkan riwayat proses pengujian yang telah dilakukan pada sistem *source detection*. Data di peroleh dari *database* yang menyimpan nilai setiap proses seperti gambar 4.39 berikut:



Gambar 4.39 *Interface* Halaman Hasil Pengujian

4.5.6.5 Rancangan menu Halaman Bantuan

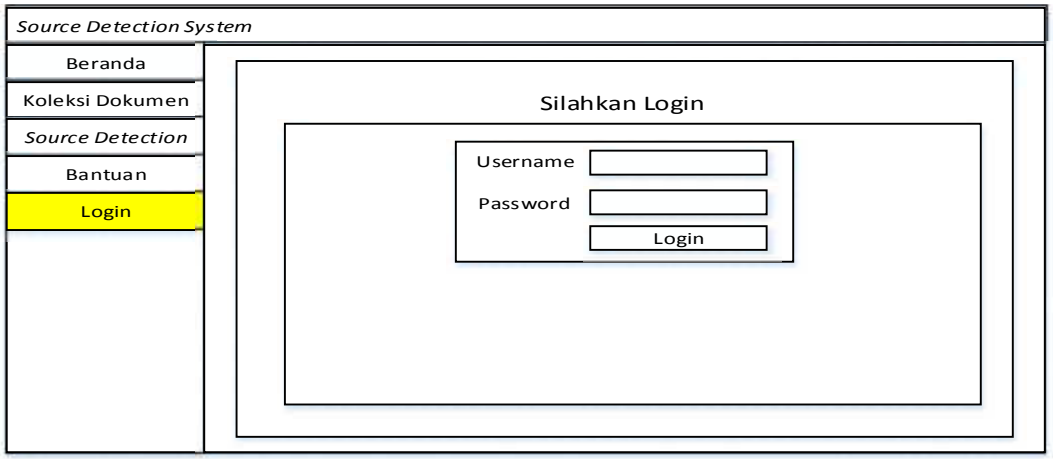
Menu Hasil Bantuan adalah halaman yang menampilkan informasi mengenai tata cara penggunaan sistem dan informasi-informasi terkait. Gambar 4.40 berikut merupakan rancangan *interface* menu bantuan:



Gambar 4.40 *Interface* Halaman Bantuan

4.5.6.6 Rancangan *Form* Login

Form Login digunakan untuk admin melakukan validasi terhadap sistem, jika data yang di berikan benar maka sistem akan mengarahkan ke halaman yang dimiliki admin dengan hak akses yang telah dijelaskan pada *Use Case Diagram*. Gambar 4.41 berikut merupakan rancangan login:



Gambar 4.41 *Interface* Halaman *Form* Login